References
BARRON, D. W. (1969). *Assemblers and Loaders*, London: Macdonald.
BASKIN, H. B. (1969). A modular computer sharing system, *CACM*, Vol. 12, pp. 551-559.
COMPUTER SURVEY (1972). Vol. 11, pp. 310-311.
GLAMORGAN POLYTECHNIC (1971). *Introduction to the IBM 1130 Assembler Language.*
IBM (1970). IBM 1130 Subroutine Library, Order No. GC26-5926, New York.
WEGNER, P. (1968). *Programming Languages, Information Structures and Machine Organisation*, London: McGraw-Hill.

Implementation problems—why are they seldom aired in the Journal?

P. Giles

Scottish Business Education Council

It is well known in educational fields that *objectives* should be established before carrying out any detailed work towards the development of a computer based system. These objectives are, of course, to be properly documented so that they may be compared with the actual outturn of events, thus improving the experience and critical ability of those directing the development of the project. In this way future developments will be better planned and implemented.

There are admitted to be problems in this approach because some benefits are not easily quantified and recent cost benefit analyses have not always produced an acceptable result—London airport proposals being a notorious example. However this is not a good reason for refusing to set objectives.

Nevertheless there are many situations where objectives are never clearly established—certainly not in print—and where, as a result, it is impossible to describe whether the development reached its objectives, diverged from them, or never arrived anywhere at all. This situation is to be deplored because it greatly inhibits the development of professional experience and judgement in the field. One major advantage of employing a software house to handle developments is that it provides an incentive to both sides to document clearly and in detail both the desired objectives and the achieved results. Why should this incentive be necessary?

To understand the reasons we must consider the subconscious objectives that are implicit in the actions of the chief protagonists in any such development. Whether intended or not, the merest suggestion of change to an established order immediately drives individuals into one or other of two social groupings. Each grouping, either *for* or *against*, may centre on a very small number of individuals. Immediately they start to group there is a subtle psychological social pressure on the remaining individuals to clarify—at least in their own minds—which grouping they belong to. This grouping tends to focus and hence to exaggerate opposition to change.

On the other hand the existence of opposition tends to heighten the ambitions and aggressive intentions of those desiring change. Any individual who has drifted to one group or the other endeavours to formalise his reasons for doing so and thus crystallises his position into a much more rigid attitude than he held previously. The problems of communication between data processing staff and users produced by this situation are well known.

However, those accustomed to managing change and development achieve their objectives more often by establishing a position of power in advance. They realise that their ability to establish this position depends largely upon their credibility and the self confidence with which they act. It is therefore essential that each development with which they are concerned shows no signs of indecision or of mistakes. The simplest and most certain method of ensuring such a satisfactory outcome is to document only achieved fact and to be ready to bury rapidly any evidence of having taken a wrong path.

This is the most practicable line of development for the non-technical manager because the academic background of many professionals inhibits them from employing such methods and leaves the field open to the experienced but unqualified manager. As such managers become more senior they become more expert in this approach and less able to see its weaknesses. Since it relies largely on a principle of competitive personal ability in contrast to the professional approach of co-operation among different specialists it leads managers into a philosophy of centralised control where delegation is reduced to a minimum and the line of responsibility must be clear. Such managers find it difficult to employ specialists and professionals to assist them because the professional approach to work and the aims that motivate specialists are foreign to their experience. Post war expansion of university and college education exaggerates this contrast to the detriment of both sides of industry.

However the blame for many unsatisfactory applications of computers belongs very clearly on the technical side. In many cases poor communication at the planning stage, or inability to visualise the resulting clerical operations in any detail, leads to the creation of very low grade clerical work in areas surrounding the computer. Any simple clerical work which is devoid of meaning to the worker, or of human interest, leads to a much higher error rate. How many system designers try to build human interest into the data preparation work? How many computerised key punch systems welcome their operators by displaying 'good morning'?

The absence of a reasonable human interface is one of the major faults of any batch processing system. What sensible human being would ever accumulate a whole day's enquiries before even starting to find an answer to the first one? If more accurate and more reliable information systems are to be set up the interface between the machine and the human being must be organised in a manner that recognises that the user is a human being and not a moron.

Magnetic stripe ledger cards go a long way towards recognising this because they enable the operator to take a knowledgable and influential place in processing the work. They also place the operator and the computer in equally important positions in so far as they enable either of them to provide an information service to the outside world. In contrast the large batch processing computer sits in a position of power from which it refuses to release any information unless the precisely correct program is supplied in the correct way with the correct job cards. What more thorough way of asserting its authority over the operator!

For a larger sum of money a full scale on-line computer system can be purchased and here the operator is in a much happier position. She is now able to provide a much more complete service to the human beings with whom she deals, and may even be able to suggest practical improvements in the existing computer responses. Furthermore she is trained to hide the computer's weakness by light conversation during the seconds it must have before it can construct its halting response.

These unquantifiable implementation problems which the computer professional may expect to meet during his career are much more difficult to solve than the technological problems more commonly expounded in the *Journal*. This note is intended to initiate further discussion on such topics so that experience of social problems may be gathered together. Perhaps it will then become better understood how to solve them, and more reports of successful fresh developments of practice can appear in print. Do you agree entirely with this problem analysis?

To the Editor
The Computer Journal

Sir

I share your mystification concerning the 'elegant trap' that I reputedly set for you to fall into (Mr. Gayler's letter, this *Journal*, Vol. 15, No. 4). Whilst I like to feel that any trap I did set would of course be elegant, in this case I must protest my innocence. The point I was trying to make (which was amply supported by Mr. Walwyn) is that the sole purpose of a computer system is to serve the needs of its users. A system may be 100% 'efficient' in its utilisation of the CPU, but if this is achieved at the expense of meeting the users' needs it is a bad system. If the only way to avoid masses of JCL is to store macros on disc, then this is a good use of the disc. (The better alternative of designing the system right in the first place is not available.)

Unfortunately, the majority of users do not realise what is possible (if they did they would not tolerate so much lousy software), and too much software is produced by designers who have never been real users. When Robert Townsend became President of Avis he insisted that all executives (including himself) should serve a spell as a counter clerk. That's what I call professionalism.

Yours faithfully,

D. W. BARRON

Department of Mathematics
The University
Southampton SO9 5NH
6 February 1973

To the Editor
The Computer Journal

Sir

Degeneracy in the Matrix of Partial Derivatives

Osborne and Watson have published in this *Journal* an algorithm for nonlinear discrete minimax approximation and an algorithm for nonlinear discrete L_1 approximation. Let $x_1, \ldots, x_m$ be the set on which approximation occurs and let the nonlinear approximation $F(A, x)$ have p parameters $a_1, \ldots, a_p$, where $p < m$. Let M be the matrix whose i, jth entry is

$$\frac{\partial}{\partial a_i} F(A, x_j) .$$

It is assumed by Osborne and Watson that M has rank p. We consider in this letter the question of which nonlinear approximations F give M this rank.

Let $\{\phi_1, \ldots, \phi_p\}$ be linearly independent on $\{x_1, \ldots, x_m\}$ and define $L(A, x) = \sum_{k=1}^{p} a_k \phi_k(x)$. By the linear independence, the matrix M of basis functions evaluated at the points has rank p. Now let ϕ be a function such that ϕ' does not vanish and define $F(A, x) = \phi(L(A, x))$, then it is easily seen that the matrix M of partial derivatives for F must also have rank p.

However, this is the only example of rank p which the author has found in a study of over twenty common nonlinear families. Consider for example the simple form

$$F(A, x) = a_1 \phi(a_2 x), \quad \frac{\partial}{\partial a_2} F(A, x) = a_1 x \phi'(a_2 x) .$$

Well known special cases are $\phi(x) = \exp(x)$ (exponential approximation) and $\phi(x) = 1/(1 + x)$ (rational approximation). If we set $a_1 = 0$, the matrix M of partial derivatives of F has its second column zero and M has rank at most 1. A similar situation occurs when we have

$$F(A, x) = \sum_{k=1}^{n} a_k \phi(a_{n+k} x) .$$

In the well known case of rational approximation, we can also have rank of M less than p. In particular when $R(A, .) = 0$, the columns of M corresponding to denominator coefficients are zero.

If the rank of M is less than p, we could perhaps have stationary points of the algorithms which are not best approximations. Consider in particular the case where

$$F(A, x) = a_1 \phi(a_2 x), \quad \phi(0) = 0 ,$$

and the algorithm is started with $A = (0, 0)$. It is easily seen that M is identically zero and the algorithms can loop.

Yours faithfully,

C. B. DUNHAM

Computer Science Department
University of Western Ontario
London
Canada
30 November 1972

References

OSBORNE, M. R., and WATSON, G. A. (1969). An algorithm for minimax approximations in the nonlinear case, *The Computer Journal*, Vol. 12, pp. 63-68.

OSBORNE, M. R., and WATSON, G. A. (1971). On an algorithm for discrete nonlinear L_1 approximation, *The Computer Journal*, Vol. 14, pp. 184-188.

Dr. Watson replies:

With the exception of those of the last paragraph, the observations in this letter are correct, although they have little relevance to the practical situation. Obviously, for the algorithms to apply it is only necessary for the matrices M defined *at the successive approximations* A^j to have rank n. The choice of an initial approximation which avoids making the rank of M trivially less than n would not appear (to me, anyway) to be a serious problem, but in the event of this happening, it would be detected by the linear programming method of solution advocated.

To the Editor
The Computer Journal

Sir

Dijkstra (1968) once wrote a letter suggesting that the GO TO statement was the principal source of trouble in debugging programs and that its elimination would be a significant step forward in programming practice and languages. In retrospect, this seemingly innocent, and to some, preposterous suggestion was the opening gun of a programming revolution that may eventually be more dramatic and far reaching than the transition from machine language to the 'higher level' languages. It seems clear now that it is feasible to abandon entirely the algorithmic concept of programming and devise new type languages based on a quite different concept of computing —that of Variable Free Programming (Backus, 1972), or Functional Programming, or generally, Non-Algorithmic Languages (NAL).

The impetus for this change seems to be the need or desire for constructing proofs of correctness of programs, and as we are discovering, proving correctness of an algorithmic like program is extremely difficult. In fact, the art of proving correctness of algorithms in general has been much neglected and relatively little is known about the subject. On the other hand, much more is known about proving theorems about or properties of functions, suggesting that if a program consisted solely of function references and that no side effects are permitted, it would be relatively easy to prove that the functions had the desired properties, and therefore, the program was correct. Thus, the desirability of functional or function oriented programs; we know how to prove things about them whereas we find it extremely difficult to prove things about programs in algorithmic languages such as FORTRAN, COBOL, ALGOL, *et al.*

In a Functional Programming Language one describes only what is to be done, not how it is to be done. LISP comes closer to being function oriented than most of the other commonly used higher level languages, yet it includes in most implementations (or aberrations) statements, labels, go to's, arrays, etc. A properly purged version of LISP would come close to a Functional Programming Language, but the implementation would have to be drastically altered in order to get run time efficiency. Furthermore, some versions of functional programming (strictly variable free programming) would exclude the LAMBDA of LISP and thus eliminate for all practical purposes even a purged version of LISP as a basis for a Functional Programming Language.

Relatively minor modifications of the ALGOL-60 syntax together with the purging of all statement like constructs, variables and side-effects could form the basis of a function oriented non-algorithmic language (called NON-ALGOL?). One also would need at least one

new type procedure, the 'procedure procedure' for declaring function valued functions and more freedom in the way procedures are used and passed to other procedures. But again, as in LISP, the implementation would require a vast overhaul to be useful.

In any event the NAL's are coming and with them, greatly simplified proof of correctness methods; whether they will sweep away the present algorithmicly oriented languages* remains to be seen. The realm of programming language research should prove quite exciting for the next 10 years while we watch the efforts of Non-Algorithmic Languages to displace the algorithmic languages and also see the competition (personality conflicts, vested interests, etc.) of the NAL's among each other for acceptance.

An international agreement or conference for standardising a NAL must eventually come to pass (like the ALGOL-60 meetings) but, it appears much too soon to effect standardisation. We need more versatility to determine just what would be the best way to structure NAL's, even though this may mean building another Tower of Babel by generating dozens of new NAL's. Also, the question of run time efficiency will play a dominant (decisive?) role. Functional or Variable Free Programming Languages are extremely difficult if not impossible to implement efficiently on existing hardware. Radical hardware revisions, possibly with extensive use of content addressable or associative memory, may be needed before NAL's are practical.

In the meantime, the defenders of algorithmic languages may yet save them by major breakthroughs in the theory of proofs of correctness for algorithms, that is, show that proving correctness of algorithms in general is not nearly as difficult as it now seems.

*As hinted by Dijkstra (1972).

Yours faithfully,

L. J. GALLAHER

Rich Electronic Computer Center
Georgia Institute of Technology
Atlanta
Georgia 30332
USA
2 February 1973

References

BACKUS, J. (1972). *Variable Free Programming*, talk at SIGPLAN Technical Session, Fall Joint Computer Conference, Anaheim.
DIJKSTRA, E. W. (1968). *CACM*, Vol. 11, pp. 147-8.
DIJKSTRA, E. W. (1972). *CACM*, Vol. 15, pp. 859-66.

To the Editor
The Computer Journal

Sir

Hashing Techniques for Table Searching

Despite the many papers on this subject, in both *The Computer Journal* (the latest one appeared in Vol. 15, No. 4, written by Messrs. Hopgood and Davenport) and other journals—particularly the *Communications of the ACM*—I have come to the conclusion that the solution which should generally give the shortest average search length is the 'common-sense' one, viz:

1. *Table size M, M prime*
   Initial entry position, P_0 = function$_1$ (key). Then if further probing is necessary, calculate Increment, I, = function$_2$ (key), such that $I < M$ and then subsequent positions, P_i, are given by
   $$P_i = (P_{i-1} + I) \text{ modulo } (M)$$
2. *Table size M, M = power of* 2
   As for (1) above, only to ensure I and M are co-prime, calculate:
   $$\text{Increment, } I, = 2 \times \text{function}_2 \text{ (key)} + 1$$
   This is the method given by F. Luccio (1972).

It seems to me that both these methods:

1. scan the entire table (as the table size and increment are co-prime);
2. minimise clustering caused by one sequence of probes becoming coincident with another sequence (because in general different sized increments are generated by different keys; although it is possible for two different keys to have the same increment, the sequences do not generally coincide until after $M/2$ probes, and so practically this coincidence has little influence on the average search length);
3. minimise clustering caused by many keys hashing to the same initial position (as their increments will generally differ).

Should anyone know of any analysis or practical work which might discredit these methods, I should be most grateful to hear from them.

Yours faithfully,

A. J. D. PAWSON

60c Hatherley Road
Sidcup
Kent
7 February 1973

Reference

LUCCIO, F. (1972). Weighted increment linear search for scatter tables, *CACM*, Vol. 15, No. 12, pp. 1045-1047.

Mr. Hopgood replies:

The major cause of the average length of search increasing when a table is becoming full is primary clustering, that is, search paths from a number of initial entry positions come together and stay together, thus creating long search paths. A second order effect is the problem of all entries from a particular initial entry position having the same search path. Mr. Pawson suggests a very sensible cure for the second-order effect which also reduces the primary clustering by having more possible search paths from each initial entry point.

The results obtained by Luccio suggest that the weighted increment linear search is comparable to the quadratic method, but no better. This implies that the amount of primary clustering remaining is about equivalent to the second-order effect. If the user feels that the second-order terms are worth removing, it is a simple matter to define a weighted increment quadratic search. Taking the average of a set of runs using random data reduces the average length of search at 90% full from 2·84 to 2·76 for the quadratic method (Luccio's method gives 2·79). As these values are never achieved in practice due to poor hash functions and non-random data, differences in the third figure are academic. In practice, I have found that the straightforward quadratic hash method is both simple to use and gives good results in a number of applications.

To the Editor
The Computer Journal

Sir

In his paper on compiler diagnostics (this *Journal*, Vol. 15, No. 4) Burgess states that it would be advantageous to standardise diagnostics for programming languages. This sentiment is often expressed but the argument does not withstand closer examination.

Would it be acceptable to allow only a standard set of diagnostics for a language? No, firstly because if the diagnostics are to be sufficiently discriminatory to be worthwhile this assumes omniscience on the part of the standards committee, and secondly because such a set of diagnostics could be applied rigorously only to an implementation of a language which had no extensions or contractions whatever vis-á-vis the language definition. Would it then be acceptable to specify a standard minimum set of diagnostics? Again no, partly because of the same problems of variations in the language but more importantly because it may hinder the development of new techniques by requiring compilers to detect a particular set of errors.

In any case, this is missing the point of standardisation. It is pertinent to standardise programming languages because these are essentially man-to-machine communication and the recipient is not intelligent. Diagnostics are machine-to-man communication and there is less need for conformity. What would help the programmer more than standardisation is a more general adoption of the practice of printing an English phrase or sentence describing an error rather than a cryptic message like 'ERROR NUMBER 8'.

Yours faithfully,

D. T. MUXWORTHY

Edinburgh Regional Computing Centre
James Clerk Maxwell Building
The King's Buildings
Mayfield Road
Edinburgh EH9 3JZ
7 February 1973

To the Editor
The Computer Journal

Sir

Your reply to Messrs Wheeler and Needham (this *Journal*, Vol. 16, No. 1, p. 18) sacrifices something of professional integrity on the altar of wit. The choice of altar is excellent, that of victim rather less so.

Certainly no editor has the right to publish under an author's name anything that the author has not agreed to publish, or to not publish anything that the editor has agreed to publish. Certainly no editor has the right to publish anything that debases standards of thought and expression. The two are not independent, but aspects of the same thing. Those who are slovenly about what they write are slovenly about what they write about. Those who read slovenly writing are pushed towards slovenly thinking.

A function of editorship is to reconcile these directives by suggesting, persuading, educating, and finally joining forces with the author to cut out the careless, the crass, and the cute. Most authors will welcome such editorial aid, if only in retrospect. Even I have done so, though admittedly a long time ago.

On the whole this *Journal* has at least an honourable record. I am probably old fashioned and ill tempered to resent a flock of unidentified OTU's (p. 30 et seq.) in the current issue. I hope, though without robust confidence, that I am joined by those who are neither in objecting to the cancerous abuse of the words 'intelligent,-ence' and (as usual) 'information'. Surely by now authors should have learnt to cross out the latter word whenever it springs to hand before mind, and substitute a more specific word appropriate to whatever is being talked about? If they have not learnt to do so, editors must teach them. That linguistic corruption is endemic does not absolve one from keeping it in check. Indeed editors could have and still can get rid of it by refusing to propagate it.

Linguistic corruption is conceptual corruption, words being used as substitutes for thought. Someone once said of computing and cognate activities that probably no profession had acquired so misleading a terminology in so short a time; certainly no profession had cared less.

Yours faithfully,
ROBERT A. FAIRTHORNE

30 Clockhouse Road
Farnborough GU14 7QZ
Hampshire
10 March 1973

Editor's comment:

I must apologise if I have allowed my natural inclination for wit to hide the meaning of my comment on the letter of Dr. Needham and Dr. Wheeler. Because of the purely practical problems involved, i.e. the amount of work, finding suitably qualified people to undertake it, ensuring that no change of meaning has taken place, it is not possible for the editorial staff of the *Journal* to monitor the standards of English used in published papers. Referees do give considerable help in this but even they are neither omniscient nor all seeing.

The Editorial Board of the *Journal* expects authors to write in concise and clear English and deprecates the use of unnecessarily long words, words of doubtful construction, and jargon. The responsibility for seeing that this is done, however, must, so long as present conditions apply, rest with the author.

To the Editor
The Computer Journal

Sir

In response to your editorial in the February 1973 issue of *The Computer Journal*, I should like to make the following comments.

It has been said of the *Journal*, that the vast majority, if not all, of the articles therein are irrelevant or incomprehensible to most Data Processing members. From my own experience, this is a fact. The stock answer in the past has been that suitable papers of a D.P. bias have not been forthcoming. I am only too ready to accept that this also is a fact.

A reason for this, I venture to suggest, may reflect my own reason, of never (before) having dared to presume that anything I could write would not become an object of derision from the august and erudite university fraternity. To put it bluntly, I and the average D.P. man are just not in their class. (If we were, then we would soon try to become one of them!)

Pleas have been made for the *Journal* to maintain 'The highest technical standards'. But how do you measure or evaluate such standards? If any such measure involves intellectual standards or advanced technical/mathematical theory, then the standards of commercial D.P. men can never rival those of the universities. Any attempt therefore to maintain the readership (or even the membership) of this 'silent' majority must inevitably involve a lowering of such standards. I do not believe that this would necessarily be an undesirable thing.

There would still be, of course, a need for the more advanced and theoretical papers, perhaps even a majority of these, but please, never forget the plain, simple and overworked commercial analyst and programmer, without whom neither the BCS nor the University Computer Departments could justify their present form of existence.

More positively, I should be very much in favour of any papers of the type suggested in your editorial, explaining the context and significance of new developments. Other areas which could materially increase the knowledge and effectiveness of D.P. members may include:

1. 'Educational' papers, outlining for instance the logic and structure of existing software. (Compilers, Operating Systems, Peripheral Control, etc.).
2. Problems of practical systems, either with solutions or merely an analysis of the problem in order to entice further papers offering solutions.
3. Systems descriptions of large or interesting applications. Preferably written *after* implementation.

With this unusual, and uncharacteristic, modesty over, I am attempting to prepare a paper on problems of correcting data errors within large integrated business systems. It will not attempt to provide answers directly, but in view of my arguments above, it will *not* achieve the intellectual standards of current articles. I should like to know if you would be interested, or am I wasting my time?

Yours faithfully,
C. R. TYLER

34 Fenton Street
Scunthorpe
Lincolnshire
28 March 1973

Editor's comment:

In the above letter Mr Tyler asks a question which, I am sure, provokes thought in many systems analysts and programmers operating in the field of commercial data processing, and, because of the lack of an answer, deters them from submitting important papers. The question is that whereas papers in the topics listed in (1), (2), and (3) above are stated in the 'Notes for authors' to be suitable for the *Journal*, what is the meaning of the phrase 'highest technical standards' when applied to those papers.

The most apt answer to the question which I have seen yet was written by one of the members of the Editorial Board. I quote:

'The criteria which I have used, when I have been asked to referee a paper, have nothing to do with academic theses; I ask myself about the paper I am to comment on,

1. is it interesting,
2. is it different from what I and others know already, a new slant on topics, etc.,
3. how does it relate to what I do know, what extensions, etc.,
4. is it correct,
5. can I learn something from the paper and is it written in a style which will enable other people to learn from it easily?'

As an editor I should add to these five questions a sixth. Is the length of the paper commensurate with the amount of new information which it presents?

What interests Mr. Tyler is, of course, what would a paper on correcting data errors have to contain in order to give an answer yes to the above questions. Assuming that no work on this topic has yet been published it should do the following:

1. Specify the class of problems which can arise in the preparation of data.

2. For each type of problem specify the possible solutions to the problem, identify the solution which has been adopted, and give the reasons for the particular choice.
3. State how the results of the investigation have been applied in the author's work situation.

Where some of the results have been presented before these should not be repeated but only referenced. Clearly, one would expect the author to be aware of such publication. In a scientific paper one would expect the author to attempt to establish a theoretical basis for his results, but this is not expected normally in a paper concerned with business applications.

I shall look forward to receiving Mr. Tyler's paper. If it does not conform to the above criteria I hope that at least I shall be able to advise him on how to amend it so that it does.

To the Editor
The Computer Journal

Sir

In Volume 16, Number 1 (Feb. 1973) R. J. Dakin suggests in his letter that 'hardware designers have so far done very little' in providing hardware address traps.

My copy of the publication 'IBM System/370 Principles of Operation' (GA22-7000-2) is out-of-date I know. However it does have about seven pages of description of the 'Program-Event Recording' facility which he may care to read. Apparently the hardware can selectively monitor for various events such as references to instructions or storage alterations in a range of locations.

Thoughtfully there is also software support for this. OS/VS2, which came out in late-72, supports it through DSS (Dynamic Support System).

Yours faithfully,

D. G. GLADING

23 Gresley Close
Four Oaks
Sutton Coldfield
Warwickshire
12 March 1973

To the Editor
The Computer Journal

Sir

The computing speed on a new machine

M. Ahmad's paper in the May 1972 issue of *The Computer Journal* attributes to us observed execution times of instructions in the MU5 computer. At the time of M. Ahmad's stay in Manchester the hardware of MU5 was not commissioned, and the times quoted must be derived from design estimates. They differ, in fact (in both directions), from our design estimates and from the (now) observed times. The instructions quoted in the paper appear to be taken from an early version of the order code, which itself differs from the implemented version.

At this stage the MU5 design team cannot endorse any detailed conclusions about the performance of MU5. A systematic investigation of its performance has been started, and the results of this will be published as soon as they become available.

Yours faithfully,

R. N. IBBETT
E. T. WARBURTON

Department of Computer Science
The University
Manchester M13 9PL
22 May 1973

Editor's comment:

It is a matter of regret that the paper of Ahmad appeared in its present form. Apart from the fact that the times produced do not have the support of the Manchester University development team, I have been informed of a number of other defects. These include a misquotation of a report of Wichmann and a failure to take account of the more recent work in this area. I hope that the Manchester team will be able to produce a definitive paper on the performance of the MU5 in the very near future.

To the Editor
The Computer Journal

Sir

The paper of Wells, 'File compression using variable length encodings' (*The Computer Journal*, Vol. 15, No. 4), comments that the decoding from variable length to fixed length '. . . if implemented entirely by software may be disastrously slow'. An extremely efficient and straightforward method of decoding is possible on a binary machine.

If the longest encoding of a symbol into a variable length code-word is of length n, then two arrays, SYMB[0:N] and NBITS[0:N] are all that is needed for decoding, where $N = 2^n - 1$. Suppose the fixed length symbol S is transformed into the code-word S' of the length m by the encoding process. To initialise the arrays for decoding, consider a binary representation of length n of the subscript k, then if the leftmost m bits of the binary representation of k are identical with S', set SYMB[k] to S and NBITS[k] to m. The algorithm for decoding is then:

A1: 1. Let k be the binary number represented by the next n bits of the encoded message.
2. The decoded symbol is SYMB[k].
3. Discard NBITS[k] from the message.
4. Return to (1) for the next symbol.

In the example of Fig. 1 of Wells' paper, both arrays would contain 16 elements;

SYMB[0] = SYMB[1] = '*e*', NBITS[0] = NBITS[1] = 3,
SYMB[2] = '*h*', . . ., etc.

It is possible to reduce the table sizes involved at the cost of added decoding complexity by assigning an encoding of only 1 bits to the most frequently occurring symbol. Suppose that in addition to the above, the most probable symbol is encoded as a string of 1 bits of length j. Then the upper 2^{n-j} elements of the arrays are unnecessary with the following more complex algorithm:

A2: 1. Examine the next j bits from the encoded message.
2. If all j bits are 1 then the decoded symbol is the most probable one and j bits are discarded from the message and return to step (1).
3. If not apply steps (1)-(3) of algorithm A1.
4. Return to step (1) for the next symbol.

In Wells' example if the symbol '*a*' is assigned a code of '11' then algorithm A2 needs tables of size 12.

Yours faithfully,

CHARLES J. GIBBONS

Mathematics Department
College of Arts and Sciences
University of Nebraska at Omaha
P.O. Box 688
Omaha, Nebraska 68101
USA
1 March 1973

Professor Wells' replies:

I am grateful to Professor Gibbons for pointing out this method which is certainly fast, with a time of order 1 compared with the time for our decoder of order equal to the average length of the code word. However, the amount of store required can grow to be very large; if p_{min} is the probability of the least probable symbol the storage requirement is in the order of $(p_{min})^{-1}$, since the longest code word will be of length $-\log_2(p_{min})$. For an extended code such as was discussed in the original paper this would require 2^{20} words of store.

Professor Gibbons does in fact point the way to a compromise algorithm, in which the most frequently occurring codewords can be detected in a single table look up into a reduced table, while less frequent codes are iteratively decoded. A note which is in preparation describing an improved hardware decoder will also include some discussion of this mixed algorithm.