Meanwhile, Schleyer et al. gave some diamantane rearrangement paths by molecular mechanics, but it is questionable whether there are other paths or not. As they rambled walking paths with many branches, they did trial calculations by molecular mechanics to find realizable paths.

If we select graph-theoretical applicants by our methods and carry out the calculations by molecular mechanics, we can predict more realizable paths with less work than using Schleyer's paths. Because even if the graph-theoretical 1,2-transmutation is possible, the chemical 1,2 shift is not always possible; however, when the former is impossible, the latter is also impossible. Therefore the shortest path which is derived by DSPA is not always a realizable path and the most favorable (thermodynamically) path is sometimes longer than the shortest path as in a tetramethylene adamantane rearrangement.[15]

We believe that the graph-theoretical methods with computer techniques developed by us would be an efficient way to study the rearrangement paths in heptacyclic triamantanes.[16]

ACKNOWLEDGMENT

This work was supported in part by a Scientific Research Grant from the Ministry of Education, Japan.

REFERENCES AND NOTES

(1) (a) N. Tanaka, T. Iizuka, and T. Kan, "A Graph-Theoretical Derivation of The Isomers of Tricyclic Hydrocarbons", *Chem. Lett.*, 539–544 (1974); (b) T. Iizuka, N. Tanaka, and T. Kan, "On the Isomers of Tricyclic Hydrocarbons C_nH_{2n-4}", *Sci. Rep. Fac. Educ., Gunma Univ.*, **22**, 47–56 (1973).

(2) T. Iizuka, H. Miura, and T. Kan, "Polycyclic Blocks and Bridged Polycyclic Compounds", *Sci. Rep. Fac. Educ., Gunma Univ.*, **26**, 79–93 (1977).

(3) (a) T. Iizuka, N. Tanaka, and T. Kan, "On the Rearrangement Reactions of Tricyclic Hydrocarbons $C_{10}H_{16}$ and $C_{11}H_{18}$", *Sci. Rep. Fac. Educ., Gunma Univ.*, **23**, 105–117 (1974); (b) "On the 1,2-, 1,3-, 1,*n*-Graph Rearrangements Corresponding to Adamantane Rearrangements", *ibid.*, **25**, 57–70 (1975); (c) "Graph-Transmutation Applied to the Rearrangement Reaction of Pentacyclo Hydrocarbons", *ibid.*, **25**, 81–89 (1976); (d) T. Kan, N. Tanaka, H. Miura, and T. Iizuka, "Chemical Rearrangement and Graph Transmutation", 26th IUPAC Congress in Tokyo, Session IV 7B408, 1977, p 1033.

(4) A. T. Balaban, D. Farcasiu, and R. Banica, "Graphs of Multiple 1,2-shifts in Carbonium Ions and Related Systems", *Rev. Roum. Chim.*, **11**, 1205–1227 (1968).

(5) A. Balabas and A. T. Balaban, "Isomerizations and Isographic Transformations", *Rev. Roum. Chim.*, **19**, 1927–1940 (1974).

(6) (a) H. W. Whitlock, Jr., and M. W. Siefken, "The Tricyclo-[4.4.0.0^{3,8}]decane to Adamantane Rearrangement", *J. Am. Chem. Soc.*, **90**, 4929–4939 (1968). (b) Smith et al. considered isomers which have two quaternary carbon atoms in adamantane rearrangement pathways: T. H. Varkony, R. E. Carhart, and D. H. Smith in "Computer-Assisted Organic Synthesis", W. T. Wipke, Ed., American Chemical Society, Washington, D. C., 1977, p 188.

(7) (a) T. M. Gund, P. v. R. Schleyer, P. H. Gund, and W. T. Wipke, "Computer Assisted Graph Theoretical Analysis of Complex Mechanistic Problems in Polycyclic Hydrocarbons. The Mechanism of Diamantane Formation from Various Pentacyclotetradecanes", *J. Am. Chem. Soc.*, **97**, 743–747 (1975); (b) E. Osawa, K. Aigami, N. Takaishi, Y. Inamoto, Y. Fujikura, Z. Majerski, P. v. R. Schleyer, E. M. Engler, and M. Farcasiu, "The Mechanisms of Carbonium Ion Rearrangements of Tricyclodecanes of Elucidated by Empirical Force Field Calculations", *ibid.*, **99**, 5361–5373 (1977).

(8) No generation of the disconnected graphs is examined in the five cyclic block graph.[2]

(9) See Schleyers's notations in ref. 7.

(10) E. M. Reingold, J. Nievergelt, and N. Deo in "Combinatorial Algorithms", Prentice-Hall, New York, 1977, Chapter 8.

(11) (a) H. L. Morgan, "The Generation of a Unique Machine Description for Chemical Structures—A Technique Developed at Chemical Abstracts Service", *J. Chem. Doc.*, **5**, 107 (1965); (b) W. T. Wipke and T. M. Dyott, "Stereochemically Unique Naming Algorithm", *J. Am. Chem. Soc.*, **96**, 4834–4842 (1974); (c) H. Miura, N. Tanaka, T. Iizuka, and T. Kan, unpublished data. We have used the other weight, prime number, rather than the extended connectivity for classification of points in a graph.

(12) As a classification of the method to find the relationships between graphs, we defined[3d] two methods called "global method" and "local method", which consist of MSPA and DSPA, and TSPA, etc.

(13) We differ between plausible and realizable; the former is mathematically possible, the latter is chemically or physically possible.

(14) H. Miura, T. Iizuka, and T. Kan, 35th Autumn Annual Meeting of the Chemical Society of Japan, 1976, Abstracts, I, 47.

(15) H. Tahara, E. Osawa, T. Iizuka, N. Tanaka, and T. Kan, "Mechanism of Tetracyclic Hydrocarbon $C_{14}H_{22}$," 1979 Winter Meeting in Hokkaido District, Abst. no. 1B05, Feb. 1–2, at Sappro.

(16) We have already applied DSPA to triamantane rearrangement, using molecular force field calculation: N. Tanaka, M. Imai, T. Kan, and T. Iizuka, "Selective Di-source Propagation Algorithm," presented at the ACS/CSJ Chemical Congress, Honolulu, Hawaii, April 1–6, 1979.

Compression of Wiswesser Line Notations Using Variety Generation

DAVID COOPER and MICHAEL F. LYNCH*

Postgraduate School of Librarianship and Information Science, University of Sheffield, Western Bank, Sheffield, S10 2TN, United Kingdom

Received October 17, 1978

The use of variety generation for reversible text compression is described briefly, and it is shown how the technique may be applied to compress Wiswesser Line Notations. The notations may be compressed, using 8-bit codes to represent variable-length character strings, to occupy an average of just under 3.6 bits per original character, an improvement of just over 55% on a fixed-length representation using 8 bits per character. This is similar to the amount of compression given by the same technique on natural language texts.

INTRODUCTION

The Wiswesser Line Notation (WLN) is widely used for storing chemical structure information in a linear form. For many applications it is a more convenient vehicle than connection tables for storing structures in machine-readable files, since it is more compact and can be input and output in a recognizable form without complex manipulations. Details of the WLN are given by Smith[1] and Ash and Hyde.[2]

When large files of WLNs are stored in machine-readable form or transmitted over telecommunication channels, it is clearly desirable that they occupy as small a space as possible. Certain contractions were introduced soon after the invention of the WLN, and can give a reduction of something like 5% in the number of characters needed.[3] However, the contraction and expansion are difficult to perform automatically, and it is often thought that the small saving in space does not compensate for the difficulties involved.

The basic WLN alphabet consists of 40 characters: the 26 letters of the Roman alphabet, the 10 digits, and the characters /, –, &, and space. (Other characters, such as asterisks in a notation for polymers, are now sometimes used, but their presence in a file makes no difference to the techniques

0095-2338/79/1619-0165$01.00/0

Table I. Character Frequencies in a WLN File

character	frequency	character	frequency
space	51954	G	4258
1	26215	L	4255
T	15002	M	4104
O	13483	2	3958
6	13179	H	3442
&	11792	-	3051
V	11686	S	2840
J	11667	X	2628
N	10304	I	2056
B	10302	3	1437
C	8158	W	1388
D	7747	Z	1060
R	7504	K	961
E	7014	P	860
5	6589	4	767
U	6041	7	579
F	5908	/	193
Q	5664	8	140
Y	5143	0	71
A	5122	9	39

Total number of characters in file = 278 561

discussed here.) Hence a fixed-length bit representation of the notation character by character requires 6 bits per character. Since many computers use 8 bits per character, a reduction of nearly 25% in the storage space required on such machines could be achieved simply by putting 4 characters into 3 bytes. To achieve more compression than this, techniques based on the statistics of occurrence of the characters can be used.

Techniques for compressing text are reviewed by Schuegraf.[4] The most important methods are Huffman coding,[5] where variable-length codes are used for (usually) fixed-length text strings, and techniques such as variety generation,[6-8] where variable-length text strings are assigned fixed-length codes. Huffman coding has the disadvantage of not removing all the redundancy in text,[9] since statistical dependence between adjacent characters is not eliminated; in addition, working with variable-length bit-strings is inefficient in computers designed to deal with fixed-length units. Variety generation overcomes the first of these problems to some extent since highly dependent strings of characters are likely to become symbols which are coded as units. If a symbol set of size a power of 2 can be chosen with the symbols occurring equifrequently, the second problem is overcome since fixed-length codes are then optimal; in practice only an approximation to equifrequency can be achieved, but it is usually a sufficiently close approximation for the extra compression obtainable by variable-length coding of the symbols not to be worth the computational complexity involved.

Tests have been carried out on methods used for text compression on a file of WLNs. The file consists of 11 601 distinct expanded notations of new compounds and compounds involved in new reactions from the 1970 issues of *Current Abstracts in Chemistry and Index Chemicus*, published by ISI. The frequencies of the 40 characters in this file are shown in Table I, and the length distribution of the notations is summarized in Table IV; the mean length is 24.0 characters. The distribution of the characters has a relative entropy[6] of 0.85, somewhat higher than the figure of about 0.75 found in natural language text.

HUFFMAN CODING

The use of Huffman codes for the compression of WLNs was suggested by Subramanian, Krishnan, and Krishnamurthy,[10] who achieved compression of a large file of contracted notations by 20–30%, presumably from a 6-bit character representation. This gives 4.2 to 4.8 bits per character, or 4.0 to 4.6 bits per character of the corresponding expanded notations. They considered the WLN alphabet as having 78 symbols, a space followed by another character being taken as a single symbol. This, of course, reflects the use of a space as a shift character, and goes a little way toward removing the dependencies between characters.

We applied a Huffman coding algorithm to the original set of 40 characters and to the set of 78 symbols used by Subramanian, Krishnan, and Krishnamurthy, using frequency statistics from our ISI file. Their method applied to our file produced 4.06 bits per (original) character, while using their codes gave 4.16 bits per character; both results are toward the better end of their range of results. Huffman coding of the original character set of size 40 gave 4.57 bits per character, confirming the advantage of using an extended character set—in this case, a very natural extension.

VARIETY GENERATION

The use of variable-length text fragments for compressing text has been described by Walker[11] (though only in the case of English Christian names), Barton et al.,[7] Schuegraf and Heaps,[8] Metcalfe and Evans,[12] and others. The method is to select a fixed set of text fragments, or symbols, and to assign these to the text in a non-overlapping way so that the text may be represented by a string of codes for the symbols. For instance, in the WLN case the notation RYR&UYVR&SR might be coded as (R)(YR&)(UY)(V)(R&)(S)(R) and SCNX1&1&1V1 as (S)(CN)(X1&1&1)(V1). It is expensive in computer time to produce the best possible coding of a given text by means of a given symbol set, but the method of longest-match assignment, which operates much faster than an optimal algorithm, gives nearly the same amount of compression;[8] with this method we assign first the longest symbol in the set which matches the beginning of the text, then the longest symbol which matches the text starting after the first symbol, and so on until the end of the piece of text is reached.

Various methods have been used for generating symbol sets to be used for compression. The algorithm used for our experiments on WLN compression is described in the next section, together with the variations used. Algorithms of this type have been used by Yeates[13] and Emly[14] for compressing prose, where they produced a representation with just under 4 bits per character.

A similar amount of compression was obtained on our WLN file; details are given below. Compression figures are given as the percentage reduction in space required, assuming an original 8-bit character representation; thus, for example, 55% compression means that on average 3.6 bits are needed per original character.

SYMBOL SET GENERATION ALGORITHM

The method begins with a symbol set containing just the primitive characters, and modifies it successively by adding new symbols formed by combining pairs of symbols occurring together commonly in the text being compressed. Such new symbols are removed from the symbol set if their frequency in the coded text becomes too low; this can occur when they are absorbed to form parts of longer symbols. The aim of the generation method is to take into account the way the symbols will be assigned.

More precisely, we consider the text to be divided up into records, such as the individual WLNs. We define a *generalized character* to be an n-gram formed from n adjacent characters in the text, where $n \geq 1$; the characters in the original text (i.e., n-grams with $n = 1$) are referred to as *primitive characters*. The term *generalized text* means a string of generalized characters representing the original text in the

sense that the latter is recovered (together with its division into records) if each n-gram is expanded to its n primitive characters. A *generalized digram* is a pair of generalized characters which are adjacent in some record of a generalized text.

We begin with a generalized text equal to the original text and a symbol set consisting of the primitive characters. This symbol set and text are modified by successive applications of the following steps:

I. A threshold frequency t is chosen.

II. The number of occurrences of each generalized digram in the generalized text is counted. That is, for each pair α, β of generalized characters, a count is made of the number of times α is the ith and β the $(i + 1)$th generalized character of some record for some i.

III. For each generalized digram whose count is at least t, a new generalized character formed by juxtaposing its components is put into the symbol set.

IV. A new generalized text is formed by using the longest-match method to assign the generalized digrams added in step III and the old symbol set to the old generalized text.

V. The number of occurrences of each generalized character in the new generalized text is counted. Any generalized character which occurs with frequency less than t is removed from the symbol set unless it is a primitive character.

VI. A new generalized text is formed by expanding each generalized character no longer in the symbol set. That is, each such generalized character is replaced by the pair from which it was constructed in step III, these being expanded further if necessary until there remain no generalized characters in the text but not in the symbol set.

These six steps are carried out as often as necessary to produce a symbol set of the size required. It is desirable for enough passes to be carried out with each threshold frequency for the process to be stabilized in the sense that the symbol set will not be altered by another pass with the same threshold. It cannot be guaranteed that the system will stabilize, or that generalized characters of length greater than two will not be produced more than once in the symbol set (for example, (ABC) could be produced both from (AB)(C) and from (A)(BC)); however these problems were not troublesome in practice, either with natural language text or with WLNs.

In the first experiments with this algorithm, the threshold was kept constant, and six or more passes were needed for the system to become stable; a threshold to produce the correct size of symbol set, or slightly above it, was found by trial and error, and low-frequency symbols were removed to give the exact size. It was found, however, that better symbol sets resulted from lowering the threshold continually, carrying out two or three passes at each threshold; the thresholds were chosen so as to add not more than 10 or 20 symbols at each pass. Further details are given in the next section.

The algorithm may be modified slightly by suppressing symbols with leading or trailing spaces. This is done in step III, by not admitting new generalized characters formed from generalized digrams whose first (or second) component is a space, for leading (or trailing) space suppression. The rationale for doing this is that space is the most common character in text and that there is less conflict between symbols to be assigned at the junctions between words if the space is put either always with the preceding word or always with the following word. This was tested for English prose by Yeates,[13] who found that either leading space suppression or trailing space suppression was beneficial, the former slightly more so. Space suppression in the context of WLNs is discussed in the next section.

Table II. Part of Symbol Set D, with Assignment Frequencies

symbol	frequency	symbol	frequency
(HJ)	576	(N D)	228
(J B1)	181	(N E)	146
(J B)	571	(N F)	234
(J C)	529	(N H)	191
(J D)	240	(N I)	140
(L3)	220	(NJ C)	282
(L4)	130	(NJ D)	211
(L5)	563	(NJ)	569
(L6TJ A)	159	(NN)	417
(L66)	457	(NV)	594
(L6V)	175	(NW)	596
(L6)	345	(O1&)	115
(L E5 B666 OV)	125	(O1)	1461
(L E5 B666)	418	(O2&)	175
(L B666)	196	(O2)	487
(L B)	159	(O&)	330
(L C)	163	(OP)	162
(M1)	193	(OSW)	189
(MV1)	254	(OTJ B1)	119
(MV)	854	(OTJ)	648
(MY)	196	(OV1)	1071
(N1&1)	308	(OV)	643
(N1&)	121	(QV)	221

Another modification, suggested and programmed by J. R. Mahoney, is to recode the original text with the new symbol set in step IV rather than coding the previous generalized text. Step VI is then replaced by a longest-match assignment of the diminished symbol set to the original text. The result should be a closer agreement between the generation method and the longest-match algorithm which will be used for assignment, and a slight improvement in compression was indeed found (see the next section).

As an example of the operation of the algorithm, consider its effect on the notation 13V1YQ1OV1. The first pass, with a high threshold, adds a few digrams to the symbol set of primitive characters, and the notation is coded as (1)(3)-(V1)(Y)(Q)(1)(OV)(1) in step IV. However, the assignment frequency of (V1) in the file as a whole turns out to be too low, because of situations, as at the end of the notation shown, where the pair (V)(1) occurs but the digram (V1) is not assigned. This symbol is therefore removed, and step VI gives the coding (1)(3)(V)(1)(Y)(Q)(1)(OV)(1). Then the coding for this notation remains unchanged for the next few passes as the threshold is lowered, until the pair (OV)(1) has a frequency above the threshold and (OV1) is added to the symbol set, giving (1)(3)(V)(1)(Y)(Q)(1)(OV1). This could result in the removal of (OV) from the symbol set if sufficiently many occurrences of (OV) were absorbed in the new symbol; in fact, this does not happen. In subsequent passes, with the threshold decreasing further, below the frequencies of the pairs (Y)(Q), (V)(1), and then (1)(OV1), the new symbols (YQ) and (1OV1) are added and (V1) returns, giving the final coding (1)(3)(V1)(YQ)(1OV1).

RESULTS

Symbol sets of size 256 were produced from our WLN file of 278 561 characters (11 601 records) by the following methods.

A. Fixed threshold, no space suppression
B. Decreasing threshold, no space suppression
C. Decreasing threshold, leading space suppression
D. Decreasing threshold, trailing space suppression
E. Decreasing threshold, no space suppression, assignment to basic text in step IV of the algorithm.

Part of the Symbol Set D is shown in Table II, together with the symbol frequencies on assignment by longest match.

For method A the threshold needed turned out to be 184; this produces a symbol set of size 258, including one repeated

Table III. Length Distributions of Symbols in Symbol Sets A to E, Together with Amounts of Compression and Relative Entropies

	Symbol Set				
	A	B	C	D	E
no. of single characters	40	40	40	40	40
no. of digrams	153	74	79	73	78
no. of trigrams	13	84	80	82	81
no. of tetragrams	41	30	31	28	28
no. of pentagrams	0	14	15	18	16
no. of hexagrams	6	7	5	10	6
no. of heptagrams	0	2	2	1	2
no. of octagrams	2	3	2	1	3
no. of nonagrams	0	0	0	1	0
no. of decagrams	1	1	1	0	1
no. of 11-grams	0	1	1	1	1
no. of 12-grams	0	0	0	1	0
(i) Compression at last pass of generation (%)	48.89	54.02	53.10	55.33	54.69
(ii) Compression from longest match (%)	50.37	54.33	52.18	55.35	54.69
(iii) Compression from longest match followed by Huffman coding (%)	52.76	57.23	55.72	58.51	57.58
Relative entropy of longest-match assignment	0.948	0.933	0.922	0.926	0.933

symbol. One of the digrams of frequency 184 was removed to give Symbol Set A. For methods B to E, three passes were carried out with a threshold of 1500, followed by two passes each with thresholds 1000, 800, 700, 600, 500, 400, 350, 300, 260, 230, 210, and down by tens to 140, then 135. After that, thresholds could be chosen to give the correct size in each case, the final thresholds being 132 for B and C, 125 for D, and 126 for E.

Table III shows the numbers of symbols of each length in the five symbol sets, together with the percentage compression given by (i) the final pass of the generation algorithm, (ii) a longest-match assignment of the symbol set to the original text, and (iii) longest-match assignment followed by Huffman coding. The relative entropy[6] of the longest-match assignment is also shown; note that there is little correspondence between this and the amount of compression.

The poor performance of Symbol Set A in comparison with the others is immediately clear, showing the usefulness of decreasing the threshold and introducing new symbols gradually; note also that Symbol Set A has an excessive number of symbols of even length, probably reflecting the large number of digrams that were introduced at the first pass.

Symbol Set D, which performs best, was generated with trailing space suppression. Symbol Sets B and E, with no space suppression, also perform well, whereas Symbol Set C, generated with leading space suppression, does not give such good compression and also shows a poor match between the generation method and longest-match assignment. This is in contrast to the situation with English prose, where either leading or trailing space suppression improves a symbol set, leading space suppression slightly more so.[13] It shows that the function performed by the space character in the text is important in determining the effectiveness of its control in symbol set generation rather than merely its high frequency; trailing space suppression in WLN symbol sets fits in naturally with its use as a shift character.

The better performance of Symbol Set E than Symbol Set B suggests that the generation algorithm works better when the assignment in step IV is to the original text. This modification would probably work even better with trailing space suppression, but unfortunately the software to do this was not readily available.[15]

Table IV. Compression Given by Symbol Set D on Notations of Different Lengths, with Longest-Match Assignments (Also Showing Number of Records in Each Range of Five Lengths, and the Numbers of Characters Involved)

length	no. of records	no. of characters	mean compression, %
1-5	28	131	15.27
6-10	783	6806	38.97
11-15	2047	26868	50.03
16-20	2470	44660	55.09
21-25	2028	46320	56.03
26-30	1529	42688	57.74
31-35	1043	34245	58.89
36-40	653	24694	58.71
41-45	398	17031	56.96
46-50	209	9999	54.68
51-55	140	7419	53.79
56-60	133	7681	51.80
61-65	44	2764	51.56
66-70	22	1497	51.70
71-75	30	2183	52.40
76-80	28	2177	48.69
81-85	7	579	45.25
86-90	5	446	52.02
91-95	4	373	42.90

Symbol Set D was also assigned, by longest match, to ten non-overlapping one-in-ten samples of the file; the lowest compression obtained was 55.19% and the highest 55.52% indicating the consistency of the compression method when applied to subfiles of the file from which the symbol set was generated.

It is interesting to notice the amounts of compression produced with notations of different lengths. Table IV shows the mean compression produced by longest-match assignment of Symbol Set D to notations of each length in the file, grouped in fives. The compression improves steadily with increasing length up to lengths between 30 and 40 characters, and then declines as the notations become still longer.

CONCLUSION

It has been shown that a variety generation method of text compression can produce similar results with WLNs to the results obtained with English prose. The similarity in the amounts of compression is surprising in view of the much higher relative entropy of the character distribution in WLNs. Notations can be stored with less than 3.6 bits per character on average, with the convenience of fixed length 8-bit codes. This represents a reversible compression by over 55% if 8 bits were originally used per character. The space requirement could be reduced to 3.3 bits per character if Huffman coding of the variety generation symbols were used, but the convenience and speed of decoding associated with fixed length codes would then be lost.

ACKNOWLEDGMENTS

We should like to thank the Institute for Scientific Information for providing us with a database, and Peter Willett both for writing programs to extract the WLNs in a convenient form and for valuable advice and comments. We are also grateful to John Mahoney and Michael Emly for suggesting improvements in the symbol set generation method.

REFERENCES AND NOTES

(1) E. G. Smith and P. A. Baker, Eds., "The Wiswesser Line-formula Chemical Notation (WLN)", 3rd ed, Chemical Information Management, Cherry Hill, N.J., 1975.

(2) J. E. Ash, and E. Hyde, Eds., "Chemical Information Systems", Ellis Horwood, Chichester, 1975.
(3) G. Vladutz, private communication, 1978.
(4) E. J. Schuegraf, "A Survey of Data Compression Methods for Non-numeric Records", *Can. J. Inf. Sci.*, **2** (1), 93–105 (1976).
(5) D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", *Proc. IRE*, **40**, 1098–1101 (1952).
(6) M. F. Lynch, "Variety Generation—a Reinterpretation of Shannon's Mathematical Theory of Communication, and Its Implications for Information Science", *J. Am. Soc. Inf. Sci.*, **28** (1), 19–25 (1977).
(7) I. J. Barton, M. F. Lynch, J. H. Petrie, and M. J. Snell, "Variable-Length Character String Analyses of Three Data Bases, and Their Application for File Compression", in "Informatics I: Proceedings of the 1st Informatics Conference, Durham, 1973", Aslib, London, 1974, pp 154–162.
(8) E. J. Schuegraf and H. S. Heaps, "A Comparison of Algorithms for Data Base Compression by Use of Fragments as Language Elements", *Inf. Storage Retr.*, **10** (9), 309–319 (1974).
(9) C. E. Shannon, "A Mathematical Theory of Communication", *Bell Syst. Tech. J.*, **27** (3), 379–423 (1948).
(10) K. Subramanian, S. Krishnan, and E. V. Krishnamurthy, "Huffman Binary Coding of WLN Symbols for File Compression", *J. Chem. Doc.*, **14** (3), 146–149 (1974).
(11) V. R. Walker, "Compaction of Names by x-Grams", in J. B. North, Ed., Proceedings of the 32nd Annual Meeting of the American Society for Information Science, 1969, pp 129–135.
(12) G. N. Metcalfe and G. B. Evans, "File Compression: a Report on BLCMP's R&D Programme", Birmingham Libraries Cooperative Mechanisation Project, Birmingham, 1977.
(13) A. R. Yeates, "Text Compression in the Brown Corpus Using Variety-Generated Keysets, with a Review of the Literature on Computers in Shakespearean Studies", M. A. Dissertation, University of Sheffield, 1977.
(14) M. A. Emly, "Compression of Continuous Text by *n*-Gram Encoding, with a Review of the Literature on Computer-Produced Concordances to Medieval Works", M.A. Dissertation, University of Sheffield, 1978.
(15) NOTE ADDED IN PROOF: This has now been done, and 55.64% compression was obtained, showing a small improvement over the performance without the modification.

A Software Controlled Data Acquisition System for Chemical Relaxation Experiments

A. M. RAO,† P. SHAH,‡ R. HAIDLE, and G. CZERLINSKI*
Department of Biochemistry, Northwestern University, Chicago, Illinois 60611

Received March 28, 1978

A system is described for the acquisition of data with a Biomation 802 transient recorder. The transient recorder is connected to a Hazeltine 2000 CRT terminal with dual tape cassette unit through a PDP 8/e minicomputer. The minicomputer stores the controlling software which allows the user to vary the sensitivity of the vertical and horizontal scale of the Biomation 802 through the keyboard of the terminal. Various simple operations are available including the storage of the data matrix on dual tape cassettes. Systems information is added to the initial data set to allow a high degree of automation in later evaluation stages. The content of the cassettes is subsequently transferred onto the disk of a large computer, utilizing a disk stored program and telephone communications. The data is then further evaluated by a sequence of programs, proceeding through several stages of evaluation. Data evaluation thus becomes highly automated and may be done much more rigorously than by graphical means or by statistical evaluations on the calculator. Since the Biomation 802 transient recorder accepts data in the form of an analogue signal and a trigger, this system could easily be used to store and process signals coming from many other sources.

INTRODUCTION

Interfacing to minicomputers was discussed previously[1-5] as well as the use of minicomputers in the handling of experimental data. Our application is somewhat different from previous ones insofar as we allow parallel transfer of the data from the Biomation 802 onto the PDP 8/e minicomputer, some preliminary editing, and subsequent transfer onto the Hazeltine tape cassette. It developed that the internal operation of this tape cassette system is such that we cannot effectively transfer more than about 1200 bits per second. The Hazeltine CRT 2000 terminal communicates with a large computer via a standard modem and phone lines at 300 band. We are also able to load horizontal and vertical scan into the Biomation 802 with the necessary information being typed in through the keyboard of the CRT terminal.

Before the use of the minicomputer, we had produced data on paper tape in rather large quantities. The rolls of paper tape were then transferred to a large computer via an overnight carrier. Unfortunately, the large volume of paper tape caused some operational problems at the main computer center. Furthermore, 10 min was required for the production of paper tape with about 1000 "points" (numbers between 1 and 256) per experiment from the Biomation 802. The conversion error of this instrument is 1 in 256 full scale (8 bit converter), adequate for most experiments. Three minutes is the optimal time between experiments for the following reasons. In our temperature-jump experiments, 3 min is required for (a) thermal equilibration after a temperature jump and (b) full attainment of a new equilibrium value after a concentration- or pH-jump. The Biomation 802 transient recorder is thus utilized on two types of chemical relaxation experiments.

The exponential decay curves may be inspected on a Tektronix 549 storage oscilloscope before conversion or on a Tektronix 602 oscilloscope after conversion. The digitized data is then transferred onto tape cassettes which are subsequently rerun under the control of a program and stored on disk at a CDC 6400 facility. Subsequently, a sequence of programs evaluates the data. Some information on the software and the details of the hardware configuration were reported before.[6] No changes were made on the Biomation 802, allowing us to use a teletypewriter and the previously employed interface (Dijiscan Model B203) as back-up for this computer system. However, we have not needed to use the back-up system since the minicomputer was first brought into operation in June of 1973. The equipment is used rather extensively in two research projects on the mechanism of action of enzymes.

METHODS

In our biochemical experiments two temperature-jump instruments are utilized, one for the detection of transmission

† Program Analyst, Humiston-Keeling & Co., 233 E. Erie, Chicago, Ill. 60611.
‡ Equipment & Software Systems, Burroughs Corporation, Detroit, Mich. 48232.
* Author to whom inquiries should be addressed at Morton 4-609, 303 E. Chicago Ave., Chicago, Ill. 60611.

0095-2338/79/1619-0169$01.00/0