Automatic Data Compression

B. A. MARRON AND P. A. D. DE MAINE
National Bureau of Standards, Washington, D. C.*

The "information explosion" noted in recent years makes it essential that storage requirements for all information be kept to a minimum. A fully automatic and rapid three-part compressor which can be used with "*any*" body of information to greatly reduce slow external storage requirements and to increase the rate of information transmission through a computer is described in this paper. The system will also automatically decode the compressed information on an item-by-item basis when it is required.

The three component compressors, which can be used separately to accomplish their specific tasks, are discussed: NUPAK for the automatic compression of numerical data, ANPAK for the automatic compression of "*any*" information, and IOPAK for further compression of information to be stored on tape or cards.

1. Introduction

Technological developments from the invention of the printing press through automatic acquisition of data from space exploration have foisted the information explosion on us. Ever-growing numbers of warehouses of data, both hard-copy records and magnetic tapes, attest to the need for somehow condensing the volume of information while preserving its content. Further, it must be possible to restore compressed information to its original form, quickly and economically.

This paper gives a general description of the multistage COPAK compressor developed for use in the *S*elf-*O*rganizing *L*arge *I*nformation *D*issemination System (SOLID System) [1–3]. The reader is directed to [2] and [3] for details of implementation.

COPAK (*CO*mbined compressor) is a *fully automatic* compressor designed to greatly reduce external storage requirements and to increase the rate of information transfer through a computer. It also automatically decodes compressed information on an item-by-item basis on demand. Prior attempts to save space [4–6] include curve-fitting, functional approximations, truncation or rounding, omission of data points, differencing, and recoding. Although COPAK uses some of these same techniques, it is unique in that it is a combination system, it permits the use of other compression devices, and it is fully automatic.

The three component compressors (NUPAK, ANPAK, and IOPAK) of COPAK can be used separately to accomplish their tasks. The *NU*meric compressor (NUPAK) achieves substantial savings in both fast and slow storage by storing numerical data in a highly compact form after its conversion to fixed point. In the truncation process superfluous bits are eliminated. The *A*lpha*N*umeric compressor (ANPAK) compresses data, either the output from NUPAK or *any* bit string, by a recursive bit-pattern recognition technique. The *I*n-*O*ut Compressor (IOPAK) stores the card image of binary-coded information on tape or cards. IOPAK is inoperative if the information is to be stored in "fast" memory.

COPAK provides the linkages in both the compression and decompression stages between these component compressors. Parity and error checks insure that machine errors in the compressed information will not escape detection. NUPAK and ANPAK have been implemented; IOPAK has not.

The algorithms described in this paper were programmed for the PILOT experiental computer at the National Bureau of Standards. PILOT is a three-address computer with 32,736 68-bit (Boolean) or 65-bit (arithmetic) words of core storage.

It is suggested that the scheme here outlined can achieve substantial storage savings for *any* data base, regardless of content, language or organization.

New components can be added to the COPAK compressor to achieve additional savings in either "fast" memory or on peripheral devices. For example, there is nothing in the design of COPAK which would prevent addition of modifications of PREST [4a], SQUOZE [4b][1] or SMARTZ (of the STRETCH system) components. Moreover, components which exploit the minimum redundancy coding techniques can be added. However, because minimum redundancy codings are based on statistical data for the information, such components would not be useful for compressing all types of information.

2. The Numeric Compressor (NUPAK)

A. PRINCIPLES OF NUPAK. Basic to the numeric compressor is the concept that all numerical data in the

* Center for Computer Sciences and Technology, Institute for Applied Technology. This work was supported by the National Science Foundation.

[1] SQUOZE and similar techniques are designed for program compression only.

RIGHTSLINK

experimental natural sciences have some limit to their significance, imposed either by the condition of their collection or by the context in which they are to be used. For example, there is no infrared spectrophotometer generally available that can measure unexpanded transmittances and frequencies to better than one percent and one reciprocal centimeter, respectively; an improved electronic thermometer has recently been described [7] which can produce results where "the intrinsic reproducibility is ±0.001°K at 77°K." Data regenerated from NUPAK's compressed form by the decoder phase of NUPAK will always be accurate to well within the limit of significance of the original data.

The following definitions are used in the NUPAK algorithm:

Suppose that there are I values of some observable x, with LS the lowest limit of significance of the values of x, and $XMIN$ the minimum value of x. Then:

bin width (BW) is one-half LS.

bin value (BV) associated with x_i is $BV_i = (x_i - XMIN)/BW$.

truncated bin value (T) associated with each x_i is computed from the BV_i value thus:

$$\begin{aligned} &\text{for } i = 1, && T_1 = BV_1 \ ; \\ &\text{for } 1 < i \leqq I, && T_i = BV_i - BV_{i-1} \ . \end{aligned} \qquad \text{(A)}$$

(Note. T is forced from floating to fixed point form without rounding.)

Depth of Representation (NDR). Further compression can sometimes be achieved by successive applications of eqs. (A), with BV_i values replaced by T_i. The depth of representation (NDR) is computed from the number of recursive applications of eqs. (A). In the programmed procedure the computer selects that value for NDR which will give the most compression.

To regenerate the original information from $XMIN$, BW, and the final T_i values obtained after recursive applications of eqs. (A), the recursive procedure is reversed to obtain new BV_i values (say $\overline{BV}_i$). The x_i values ($\bar{x}_i$) are computed thus:

$$\bar{x}_i = (\overline{BV}_i + 0.5)BW + XMIN \qquad \text{(B)}$$

Since $|\bar{x}_i - x_i| \leqq BW/2.0 = LS/4.0$, the regenerated values ($\bar{x}_i$) are well within the "lowest limit of significance" (LS) of the original values (x_i). The term 0.5 in eq. (B) halves the maximum error which can occur in truncating the bin values.

B. Structure of Compressed Information. For illustrative purposes, assume that the computer used has N-binary bits per arithmetic word, and that no T_i value to be stored contains more than $(N/2 - 4)$ bits. The truncated bin values T_i with depth of representation NDR are calculated as described above, and then compressed into array A in the following way:

The leftmost six bits of each word of array A contain three items (S, F, and NS) which define the amount of information and type of compression in the word and whether the next sequential word contains further T_i information. In the encoding procedure the computer automatically deduces those S, F, and NS values which give the greatest compression. In the fully automatic decoding procedure this information (S, F, and NS) is used by the computer to expand the compressed information to its original form.

S (*one bit*) indicates whether or not this is the last word in the array.

F (*one bit*) indicates whether the T_i values stored in this word are stored sequentially, or in a form indicating repetitive equal values.

NS (*four bits*) is the number of equal segments in this word. It is automatically computed from the available space ($N - 6$ bits), the size of the T_i values, and a consideration of the alternate ($F = 0$ or 1) forms of compression. *The leftmost bit in each segment designates the sign of the item.*

Table I illustrates the structure of compressed information. Note especially that the three lead words in the encoded output contain all the information that is necessary for the computer to *automatically* decode and check the information.

C. Example of Use of NUPAK. To demonstrate the actual savings in storage that can be achieved with NUPAK, 561 data pairs from the standard polystyrene film infrared spectrum were compressed to 68 PILOT words. (The 561 transmittance values were compressed to 63 words; the 561 regularly spaced frequency values were compressed to only 5 words.) Considerably less than one second of PILOT machine time was required. Decompression was faster.

3. The Alphanumeric Compressor (ANPAK)

A. Principles of ANPAK. In the SOLID System, input to ANPAK is either compressed numerical information from NUPAK, or the original input information if compression via NUPAK failed or was not called upon. It should be noted that ANPAK is a recursive bit-pattern recognition technique and is therefore *linguistically independent.* Further, because the original bit-pattern can be regenerated, there is no loss of information in ANPAK.

The following definitions are used in the ANPAK algorithm.

For illustrative purposes, it is assumed that a string (T) of $N2$ bits is to be compressed:

A *code* contains 2^{CW} code words, each with CW bits. $N1/CW$ must be an integer, where $N1$ is the number of bits in the machine word.

A *lexicon* (TL) discloses which of the 2^{CW} code words have been used to achieve compression, and in what manner.

A *cord* (CD) contains R consecutive code words in the string T. R is a positive integer; $N3$ ($= R \cdot CW$), the number of bits in the cord, cannot exceed $N1$.

RIGHTSLINK

TABLE I. Illustration of Encoding Procedure in the NUPAK Algorithm

Input consists of SOS (= 0), JI (= 18), LS (= 0.020) and eighteen x_i values. $XMIN$ (= 1.010) and BW (= 0.010) are used to calculate the BV_i; then these are used to compute T_i, and NDR. The final information (SOS, NDR, $XMIN$, T_i, and BW) is stored in six 65-bit computer words. NDR, SOS, S, F, and NS are codes automatically derived (and in decoding executed) by the computer.

x_1	BV_1	T_1 NDR=2	T_1 NDR=3	x_1	BV_1	T_1 NDR=2	T_1 NDR=3
1.010	0.0	0	0	35.010	3400.0	- 200	- 300
2.015	100.5	100	100	40.011	3900.1	500	700
4.011	300.1	200	100	43.012	4200.2	300	- 200
7.013	600.3	300	100	50.011	4900.1	700	400
11.014	1000.4	400	100	70.010	6900.0	2000	1300
16.011	1500.1	500	100	71.015	7000.5	100	-1900
22.010	2100.0	600	100	66.011	6500.1	- 500	- 600
29.011	2800.1	700	100	80.012	7900.2	1400	1900
37.012	3600.2	800	100	80.013	7900.3	0	-1400

Step IV (Encoded Output)

SOS =-NDR = -3

BW = 0.010

XMIN = 1.010

T_i values are stored in 3 words from high order to low order address.

(Decimal T_i are shown)

S	F	NS	Segment 1	Segment 2	Segment 3		
1	1	3	-600	1900	-1400	0	0

65,64,63- 60,59 41,40 22,21 3,2,1

S	F	NS	Segment 1	Segment 2	Segment 3	Segment 4		
0	1	4	-200	400	1300	-1900	0	0	0

65,64,63-60,59 46,45 32,31 18,17 4,3,2,1

S	F	NS	Segment 1	Segment 2	Segment 3	Segment 4	Segment 5			
0	0	4	0	100	8	-300	700	0	0	0	0

65,64,63-60,59 49,48 38,37 27,26 16,15 5,4,3,2,1

Bit structure of Segment 4 in the last word:

-300

1	0	1	1	0	0	1	0	1	1	0	0
11	10	9	8	7	6	5	4	3	2	1	

A *bit-map* (BM) for one of the 2^{CW} code words discloses the positions of the code word in the string T. Terminal zeros in a bit-map are omitted; e.g., for T = 10101101010101010001000, the bit-map for 101 is 1001.

In *type I compression*, an unused code word is substituted for a cord.

In *type II compression*, code words are removed from the string, and their locations designated by bit-maps. Briefly, the compression procedure is as follows:

A string is *irreducible* if compression cannot be achieved.

TABLE II. Illustration of the Irreducible String Assembly in ANPAK

Fixed length fields are marked with an asterisk (*). One machine word with the State of System commands (SOS) preceeds the string

*	B	The number of bits in the irreducible string.	
	I	The irreducible string.	
*	ND_i	The number of composite code words in the lexicon associated with CW_i.	
*	CW_i	The number of bits in the code.	
	ACW_{ji}	jth code word associated with CW_i	A typical lexicon entry for Type I Compression
*	R_{ji}	Number of code words in CD_{ji} ($R_{ji} > 1$)	
	CD_{ji}	The cord which was replaced by ACW_{ji} to achieve compression	
	ACW_{ki}	kth code word associated with CW_i	A typical lexicon entry for Type II Compression
*	R_{ki}	$R_{ki} = 1$	
*	NB_{ki}	Number of bits in the bit map associated with code word ACW_{ki}	
	BM_{ki}	Bit map associated with code word ACW_{ki}	

Step I. The initial value of CW is set on input. For alphanumeric information CW is set equal to six or eight (e.g., BCD or ASCII code).

Step II. The lexicon associated with code CW is constructed as follows. An array of 2^{CW} machine words is cleared to zero. For each code word in string T the appropriate word in the array is incremented by one. The unused code words are stored in a new array for use as substitutes for repeating cords in the string.

Step III. The highest value of R is selected. (In a word-oriented computer, this value depends on $N1$).

Step IV. A counter, K, is set equal to zero.

Step V.

a. Cord, CD_{N3} (where the subscript denotes the length of the cord) is set equal to bits $(K \cdot CW + 1)$ to $(K \cdot CW + N3)$ in string T.

b. A search of string T with CD_{N3} discloses whether compression can be achieved. If $R > 1$, compression is achieved by substituting the first unused code word in TL for CD_{N3} wherever it occurs (Type I Compression). If $R = 1$, a bit-map for the code word is constructed and the cord is removed wherever it is found (Type II Compression). See Section 3B for the structure of the compressed information.

c. If compression was achieved, repeat from Step IV with the compressed string. *If all the unused code words* have been assigned, set $R = 1$ and return to Step IV. If no compression was achieved, K is incremented by one and control goes to Step Va. If all cords CD_{N3} in T have been examined, control goes to Step VI.

RIGHTSLINK

Step VI. R is decremented by one. If $R \geqq 1$, control goes to step IV. It $R = 0$, control goes to step VII.

Step VII. If no compression was achieved with CW, it is incremented by ones until $N1/CW$ is again an integer. If $N1 = CW$, the compression procedure is complete. (See Section 3B for structure of compressed information.) Otherwise, control goes to Step II. NAP (see Figure 1) indicates whether or not alphanumeric compression occurred.

B. Structure of Compressed Information. The compressed information with its lexicon and all of the information necessary for decompression, consists of a single, compact, *self-defining* string with a mixture of fixed and variable-length fields. The structure of the string is illustrated in Table II.

The decompression procedure uses the fixed field entries in the string to control the expansion of the string to its original form.

C. Example of the Use of ANPAK. As a demonstration of compression with ANPAK, the first four pages of the draft of [3] were compressed. The original string consisted of 15,180 bits. The compressed string, including its lexicon and commands for automatic decompression, consisted of 9,291 bits. This represents a savings in storage of 39 percent.

4. IOPAK and COPAK

The *I*n-*O*ut compressor (IOPAK) will be closely patterned on the University of Illinois binary compressors[2] which packs twenty machine words per card. When it has been implemented, IOPAK will store the card images of binary-coded information on tape or cards. In COPAK, compressed information from ANPAK, or, if no compression occurred there, either the compressed information from NUPAK or the original input is automatically processed by IOPAK before it is stored. IOPAK is inoperative if the information is to be stored in core or on disks.

Figure 1 illustrates the interaction of the components of COPAK. Note especially that the system commands SOS, NAP, and NDR (see [2] and [3] for their definitions) which are stored with the information, *automatically* control both the compression and decompression phases of COPAK.

5. Further Comments

The *CO*mbined compressor (COPAK) with its three fully automatic components (NUPAK, ANPAK, and IOPAK) significantly reduces both fast and slow storage requirements. NUPAK is activated only when numeric information is to be compressed. ANPAK can compress any bit string without loss of information. IOPAK, not

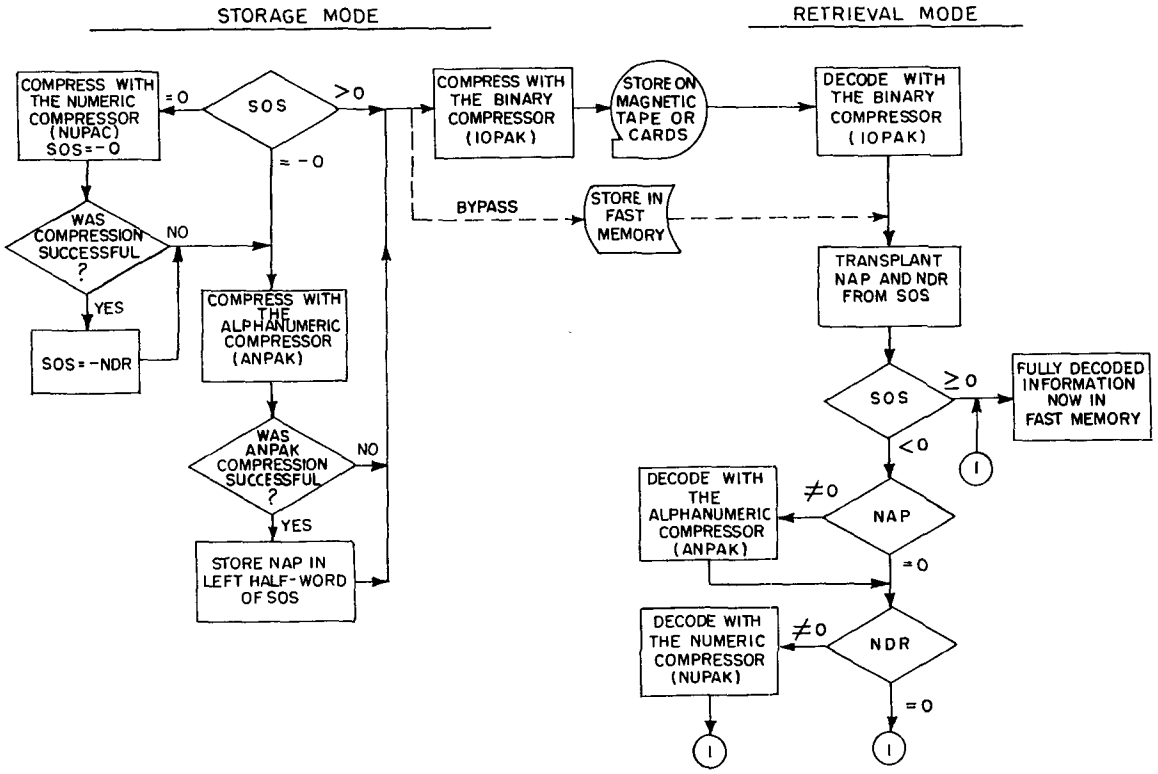

Fig. 1. Schematic flowchart showing the inter-relationships among the three components (NUPAK, ANPAK, and IOPAK) of the numeric-alphanumeric-binary compressor (COPAK) in the storage (encoding) and retrieval (decoding) modes. NAP and NDR are computed in the storage mode. Note that IOPAK is only operative for slow storage (magnetic tape, cards). COPAK is assumed to be called in by a monitor which specifies the mode.

[2] BININ and BINOUT routines at the University of Illinois.

RIGHTSLINK

yet implemented, would further reduce slow storage requirements. The present form of ANPAK and NUPAK demonstrate the feasibility of the technique, but they have not yet been optimized.

We suggest that these compressors would be applicable to such diverse situations as storage of full text, storage of large masses of experimental data, and transmission of television patterns. They are economically feasible since read-in time for compressed information plus time for decompression is significantly less than read-in time for the original information.

The method described represents an advance over earlier methods by virtue of its being fully automatic.

RECEIVED AUGUST, 1966; REVISED MARCH, 1967

REFERENCES

1. DE MAINE, P. A. D., AND MARRON, B. A. The SOLID System I. A method for organizing and searching files. In George Schecter (ED.), *Information Retrieval—A Critical View*, Thompson Book Co., Washington, D.C., 1967.
2. ——, KLOSS, K., AND MARRON, B. A. The SOLID System II. Numeric compression. Nat. Bur. Standards, Tech. Note 413.
3. ——, MARRON, B. A., AND KLOSS, K. The SOLID System III. Alphanumeric compression. Nat. Bur. Standards, Tech. Note 413.
4a. PREST deck in the IBSYS System. IBM 7094 Manual;
4b. BOEHM, E. M., STEEL, T. B., JR. The SHARE 709 System. *J. ACM 6* (1959), 134.
5. Use of differences to obtain functional values—application of machines to differencing of tables. *J. Am. Stat. Assoc. 41*, (1946), 233-237.
6. MYERS, W., TOWNSEND, M., AND TOWNSEND, T. Data compression by hardware or software. *Datamation* (April 1966), 39-43.
7. Standards and calibration. *Tech. News Bull.* Nat. Bur. Standards, Mar. 1966, p. 45.

A Computer System for Inference Execution and Data Retrieval

R. E. LEVIEN
The RAND Corporation, Santa Monica, California
AND
M. E. MARON*
The University of California, Berkeley, California

This paper presents a RAND project concerned with the use of computers as assistants in the logical analysis of large collections of factual data.

A system called the Relational Data File was developed for this purpose. The Relational Data File is briefly detailed and problems arising from its implementation are discussed.

1 Introduction

The computer's potential value as an assistant in the logical analysis of large collections of factual data was clearly and explicitly recognized soon after construction of the first automatic sequence-controlled calculator. Vannevar Bush wrote in his famous *Atlantic Monthly* article [1]:

> The repetitive processes of thought are not confined . . . to matters of arithmetic and statistics. In fact, every time one combines and records facts in accordance with established logical processes, the creative aspect of thinking is concerned only with the selection of the data and the process to be employed and the manipulation thereafter is repetitive in nature and hence a fit matter to be relegated to the machines. . . .
>
> The scientist . . . is not the only person who manipulates data and examines the world about him by the use of logical processes. . . . Whenever logical processes of thought are employed—that is, whenever thought for a time runs along an accepted groove—there is an opportunity for the machine.

Such opportunities for the machine are myriad. Two will serve to illustrate the promise.

A *corporate data analysis system* would contain a computer store of factual data about the products, markets, facilities, finance, plans, and personnel of the firm. Corporate managers would employ it to answer questions of fact and as an assistant in estimating consequences, testing alternatives, and drawing inferences.

A *scientific activities data exchange* would comprise a large store of factual data about the scientists, projects, publications, organizations, conferences, and contracts associated with research in a particular field of science. To scientists, administrators, and editors it would be a source of data and an assistant in drawing conclusions about relevant publications, interested researchers, competent organizations, and important trends. Unlike literature searching systems, it would also enable specific information about persons, organizations, and projects to be retrieved through searches of a wide variety of factual data about the scientific context.

The promise is there. What of the progress? In the twenty or so years since Bush wrote his article, computers have frequently been used to store, select, and manipulate many kinds of data. And in the last decade efforts have been made to mechanize the execution of logical analysis as applied to theorem proving and problem solving. Nevertheless, the application that Bush envisaged is yet to be realized. The combination of a large data base and the tools of logical analysis in a single system has not been provided. In fact, it has been tried only rarely.

In 1962 Manfred Kochen proposed the AMNIPS system [2], which did combine storage of a large body of

This research is sponsored by the U.S. Air Force under Project RAND—Contract No. AF 49(638)-1700.
* School of Librarianship

RIGHTSLINK