SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 6, 397–403 (1976)

Indexing with Coded Deltas—A Data Compaction Technique

M. VISVALINGAM
Census Research Unit, Department of Geography, University of Durham, England

SUMMARY

The paper describes the coded delta scheme, which is one of the methods used by the Census Research Unit, University of Durham, for compacting the 1971 U.K. census data. It evaluates the merits and limitations of the technique in relation to the characteristics of the data set and other techniques available for compact encoding of numeric and string data.

KEY WORDS Compaction Sparse data Strings Coded delta Bitmap

INTRODUCTION

There has been a progressive trend in the social sciences to collate and process larger and larger data sets so that they form an expanding data base for several applications. The grid square-based 1971 census data for the U.K. is one example of a large data base, with which the Census Research Unit (CRU) of the Department of Geography, University of Durham, is investigating complex methodological issues in the mapping and spatial analysis of population characteristics. Problems of scale relate not only to the effect of variations in size of the areal unit on various demographic variables, but also to the assessment of the virtues of constant spatial versus constant population units. A series of demographic types and regions will be identified by means of multivariate statistical analysis and computer mapping of national and regional data will be carried out. While in theory most of these problems are programmable on the computer, severe practical problems arise because the volume of census data for kilometre square units alone exceeds 157 million words: the requirements for various applications involve complex search mechanisms and selective access to sections of the data. Thus the limiting factors that decide the practicality of solving given problems are the availability of core, backing store and execution time. An information storage and retrieval system, which would also preprocess the data for other statistical and survey analysis packages and undertake a certain amount of processing and mapping functions itself, is currently being designed. Besides the design of suitable data and file structures, one of the most basic operations is to compress the vast body of data into as little disk space as possible—not only to conserve space but also to reduce transfer time. Different methods of data compaction were employed based on the structure and other characteristics of the data body.[1, 2] Suppression of zeros was one of the processes by which the file was compacted to half its previous size. This paper compares the performance of the conventional bitmap scheme and the coded delta scheme (which takes advantage of an inherent property of the census data) for indexing non-zero value elements. The suitability of the coded delta scheme for other applications is also suggested.

Received 28 July 1975
Revised 20 October 1975

BACKGROUND TO THE DATA

The Office of Population Censuses and Surveys (OPCS) provides Small Area Statistics, consisting of 1,571 population, household and socio-economic variables.[3] Individual household returns are aggregated into larger spatial units, including the British National Grid squares. The population and household data for Durham census county comprises 920 variables for each of the 2,041 1-km square units populated in April 1971; unmodified this occupies 1,968 pages on an IBM 2314 disk pack. (A page is 4,096 bytes or 1,024 × 32-bit full-word units. In the Michigan Terminal System (MTS) used on IBM 360 and 370 computers shared by the Universities of Durham and Newcastle, the size of the file includes overheads for file organization. For sequential data files this overhead represents 16 bytes for the file header and 6 bytes per record.) In addition to the population and household data some 651 socio-economic variables are available for 1,275 grid squares. The total amount of data for some 100,000 grid-based units in the U.K. is estimated to be in excess of 600 million bytes—in the form made available by OPCS and transformed into IBM computer 4-byte words. Unmodified this would require at least 14 magnetic tapes (9-track tapes 2,400 ft long with a recording density of 1,600 bpi) or at least six single density 3,330 disk packs, each with a capacity of 100 megabytes. As the mounting and dismounting of magnetic tapes and disks by human operators is time-consuming, especially when the user is on-line, it is necessary to pack as much information as possible into each backing store unit, to avoid having to read several disks to extract the required information.

Experiments were conducted with the 100 per cent coverage population data, consisting of 471 variables for Durham census county, to select the most suitable methods for compaction and to ascertain the degree of compaction that could be achieved. This population data alone initially occupied 1,006 pages. The processes of data culling[4] to remove redundant information, the freeing of storage allocated to data that is suppressed owing to confidentiality restrictions and trimming the size of the data elements contracted the file to 249 pages.[2] However, as the data body contained only about 36·5 per cent non-zero value elements, the file could be further compressed with indexing algorithms conceived for the rapid and efficient processing of large order sparse matrices. Pooch and Nieder[5] in their survey of indexing algorithms defined a sparse matrix as one possessing about 15–25 per cent non-zero values. Of the conventional methods that they and Reid[6] discuss, only the bitmap scheme was likely to be of any relevance as the Durham data is not sparse by their definition. Since the position of the data item in the OPCS record identifies the variable, several applications demand that the array be restored back to its original form for in-core processing. Hence, the regeneration of the original array was used as the criterion for comparing the execution time for the different indexing schemes. The bitmap scheme further reduced the file to 121 pages (Table I) but it was found to be relatively slow in our applications as the distribution of non-zero elements often requires that the whole bitmap be

Table I. Comparison of bitmap and coded delta schemes for zero suppression of 100 per cent population data for Durham census county (refer to text for explanation)

	Prior to zero suppression	Bit encoded	Negative deltas
Size of file (pages)	249	121	124
Execution time to regenerate original file (sec)	11·4	26·2	18·4

tested. Alternative schemes of delta indexing were also considered. The coded delta scheme, which is a new variation of the method of run-length coding, proved to be substantially faster than the bitmap scheme.

DELTA INDEXING

The bitmap scheme

In this scheme only the non-zero value elements are stored and their position is indexed via a bitmap consisting of 0's and 1's for the zero and non-zero value elements respectively. For example, the array of N (= 12) values:

(0, 0, 0, 3, 0, 5, 0, 0, 0, 0, 23, 0)

would be stored as the reduced vector of n (= 3) elements:

(3, 5, 23)

and indexed with the bitmap:

(000101000010)

Even if the total number of non-zero elements (n) is retained to minimize the number of tests involved, restoration of the array requires that the bitmap be sequentially indexed until the last set bit. Storage overheads of the bitmap scheme are constant and amount to:

$(N/b)+1$ elements

where b = the number of bits allocated to each element. For the population data this amounts to 31 half-words per grid square.

The delta indexed scheme

The bitmap scheme was then compared to the coded delta method, which is a modification of the methods of run-length coding (Rosenfeld,[7] p. 16) and the delta indexing scheme.[5] With reference to picture processing Rosenfeld pointed out that when there is a tendency for one value to be more probable than all the others put together, the highly probable value tends to occur in 'runs' and the others in isolation. He suggested that it may be more economical to encode the first value of each run and the length of the run (giving rise to the term 'run-length coding'), or the first value of each run and its position in the sequence of values, rather than encoding every value in sequence. A similar scheme of address mapping is widely used in the indexing of sparse matrices, when only the non-zero values are stored together with their *absolute position* in the array. In large matrices, the co-ordinates defining the position of non-zero values are also likely to be large. The delta indexing scheme minimizes these storage requirements, by encoding with each non-zero value element a delta, which is the difference in position between the corresponding and preceding non-zero elements. The difference, being smaller in value, requires a smaller unit of storage for the elements of the indexing vector than does the absolute position. The position of the non-zero element is then determined from the sum of the preceding deltas.

Delta indexing is substantially faster than the bitmap scheme for processing of non-zero values. While the bitmap scheme can involve a test of the entire bit matrix, the number of iterations in delta indexing is equal to the number of non-zero values, i.e. the matrix density. However, as the number of deltas is equal to the number of non-zero values, the method is very demanding on store when a matrix is not very sparse. In the Durham

population data, with an average matrix density of 36·5 per cent, this requires 129 indexing units for a record with 473 elements. Hence the coded delta scheme was evolved with a view towards retaining a faster scheme of indexing while cutting down on core requirements.

The coded delta scheme

The coded delta scheme eliminates the need for an indexing vector by encoding the position and number of zero values in the 'same vector as the non-zero values'. The coded delta is a single value that gives the number of zero elements which separate two non-zero ones. The delta is encoded within the reduced vector, replacing *at least* one zero value, by using a single bit to differentiate it from the true matrix values. As the census data consists of counts which are always zero or positive (a feature present in several types of numeric social science data) the topmost (sign) bit can be used to 'flag' a delta, which takes the form of a negative number. For example, the previous array of 12 values can be encoded as:

$$(-3,\ 3,\ -1,\ 5,\ -4,\ 23,\ -1)$$

The trailing delta like the trailing zero bits in the bitmap is significant if backward processing is also desired. Owing to the method of coding deltas only when zeros do occur, this method never exceeds the storage requirements of the original arrays, no matter how dense the arrays. The savings made depend entirely on the number of deltas (d) required, i.e. on the number of blocks of intervening zeros. Hence in terms of store the coded delta scheme breaks even with the bitmap scheme when the requirements of the bitmap equals the number of blocks of zero value elements. With the Durham population data, the coded delta scheme becomes relatively inefficient in terms of store when the number of blocks of zero value elements exceeds 31. Compaction via the coded delta scheme incurred 3 pages more than the bitmap scheme (Table I), but it proved to be a considerably faster method of indexing. While the algorithm for bitmap indexing requires no further operation for the zero value elements it can involve a test of the entire matrix; close to N iterations may be required. With negative coded deltas, while there is an additional execution for the zero-value elements the number of iterations is only $n+d$; as the compacted file is less than half its previous size this is likely to be closer to $N/2$.

The recognition of a delta by its coding rather than its position allows a more economical and versatile packing scheme. Where matrix elements require a very small storage unit, for example 4, 6 or 8 bits, the value of a delta may exceed the unit's capacity. Pooch and Nieder point out that if non-zero elements of 8 bits are farther apart than 255 columns, then a greater number of bits must be allocated for each delta in the delta indexing scheme or the method must be abandoned. But the coded delta scheme copes with the occasional large counts more neatly by coding the overflows as *successive deltas*; the indexing algorithm should take the sum of any number of successive deltas. It must be appreciated however that, being a species of run-length coding and delta indexing, the coded delta scheme shares the disadvantage of not allowing direct access to individual elements in the middle of a compressed sequence. Moreover it does not even have the bitmap's capability of ascertaining the zero/non-zero status of individual elements in a random fashion.

OTHER APPLICATIONS

1. Coded deltas may be used in other applications, taking advantage where desired of machine-dependent features. Negative coded deltas work equally well when the non-negative values are represented in floating point form, for example for ratios,

percentages and proportions. When integers are of any sign, either the top or the bottom bit of the storage unit (coding +/− and odd/even states respectively) may be reserved for differentiating the deltas. This however reduces the capacity of the unit by half and requires an additional shift operation. It may be convenient to transform the values to positive ones by adding a suitable constant. For example, values from −15 to 20 can be represented to occur from 0 to 35.

2. Amidon and Akin[8] discussed several methods for compressing spatial data on land use. Collected into a uniform grid, these form a matrix filled with character codes, representing differing land use types (map conditions), which generally occur in clusters. Methods based on Rosenfeld's[7] run length coding reduce the original matrix to a vector consisting of pairs of elements (n, c) where n = the number of repeats or the length of the character code, c. Both n and c are coded in BCD form. The principal modes adopted are uniform and variable compressions, which use a fixed or variable number of BCD digits to code the length. For example, a reduced vector from a two-digit uniform compression takes the form:

(11a03d14z02f...)

and that resulting from variable compression takes the form:

(11a/3d/14z/2f...)

The underlined characters in the above examples are the feature codes, while the numeric characters indicate the length. Variable compression was considered more efficient when lengths vary widely. The slash mark represents the additional character needed to separate the lengths.

Amidon and Akin considered the possibility of suppressing slashes but did not pursue this further because it requires that the character for lengths be differentiated from those for codes, thereby reducing the capacity to store map conditions by a third. In this problem, coded deltas offer a fast and compact scheme, especially if one considers machine-dependent features in the selection of map codes so that deltas can be recognized by the state of the most strategic bit. To retain BCD alphanumerics in an IBM environment, one only needs to test B_0 of the byte to recognize the positive binary-integer deltas, each of which can index 127 repeat codes. The method of encoding not only makes slashes redundant but also avoids having to store feature codes when overflows occur by coding them as successive deltas. Compaction by the coded delta scheme would reduce the previous example to:

(11a3d14z2f...)

Each underlined value is a binary integer represented by a one-byte delta. Hence the lengths require half the previous requirements and since they are held on external storage in the more compact internal format, no conversions will be necessary before indexing. As a further consequence of the shortness of the compacted data strings, fewer tests are required, making the method in general more efficient.

Feature codes may, however, require two or more characters. There are two alternative solutions for the two-character codes used by Amidon and Akin. If the lengths are frequently in excess of 127 it may be more convenient to allocate 2 bytes to both n and c. Alternatively, a more compact packing scheme similar to that used by compilers for storing literals and constants could be adopted. For example, the compacted string resulting from two digit uniform compression[7] take the form:

(v29911vvv199v55vv36vv75vv66vv199v66v...)

in which the pair of underlined characters comprise the feature code. The symbol, v, represents the space character. This could be more effectively compacted by storing each possible feature code only once in a separate string, for example:

(99, vv, 5v, 6v, ...)

These are indexed in correct sequence and generated to the required length by another vector, consisting of positive pointers(r) which reference to feature codes, and negative deltas (d) which indicate the length of the run. For example:

$(-2, 1, -11, 2, -1, 1, -5, 3, -3, 4, -7, 3, -6, 4, -1, 1, -6, 4, \ldots)$

Each single one-byte delta encodes a run length of 128 repeats, while a single one-byte (r) in a sequence can index the most frequent 127 feature codes. Greater lengths and many more codes can be indexed by coding successive (d's) and (r's) respectively and using the sum of the overflows. By identifying different types of components with a one-bit flag, the syntax and storage allocation for units of the index vector can be modified to suit the particular application.

3. A scheme similar to the above may be used to minimize storage allocation for textual data. Mayne and James[10] factorized common strings and stored them in a separate table. These were identified by unique two (or more) character codes, the first character being a symbol which does not occur in the uncompressed text. Wagner[9] had used a more efficient scheme for reducing the total amount of space used by the PL/C compiler for diagnostic messages. He identified common phrases by a two-character flag consisting of the character 'P' and a unique number. He similarly identified the residual text by the code 'C' and a number which indicated the number of intervening characters between phrase references. This eliminated the need for scanning through the text for the occurrence of 'P', which thereby need not be unique. As the major space overhead is for the flags (P and C) and (number), the cost can be halved if the number itself flagged its identity by a one-bit code. Each positive number would transfer 128 characters while a single negative number can be used to index the most frequently occurring 128 phrases. The absolute value of the sum of a run of negative numbers allows indexing of many more phrases. The storage overhead can be considerably reduced when there is a frequent alternation between phrases and character strings.

CONCLUSION

When values have a tendency to cluster within a data set, overall storage requirements can be reduced by a method of run-length coding called the coded delta scheme. The differentiation of the values from the deltas *via* the state of the coded bit requires that the lengths be included only when repeats do occur. This produces a more compact list than is possible with schemes that identify lengths of references by their position within the vector (or an associated vector) or by the inclusion of separate flags.

A high degree of compaction is achieved at some cost. Although the method lends itself to backward processing, the reduced vector has to be processed sequentially. Also as one bit of the storage unit flags the delta, the capacity of the unit is reduced to half its normal capacity.

With numeric data the method is quite useful when the frequency of occurrence of the modal value is not overwhelming enough to justify other conventional methods of indexing sparse matrices. The Durham population data file (with an average matrix density of 36·5

per cent) was compressed to half its previous size, using the coded delta scheme. While the absolute space savings are not enormous a similar 50 per cent reduction in the U.K. file will be well worthwhile. The coded delta scheme was substantially faster than the bitmap scheme for regenerating the original array of values although it was slightly less efficient in terms of store.

The method is also being applied to data that consists of sets of attributes. When sorted into some spatial order, these tend to exhibit a higher degree of clustering. Coded deltas can also be incorporated into the syntax of other kinds of data when economy of storage is a crucial issue.

ACKNOWLEDGEMENTS

The author is extremely grateful to Dr. D. W. Rhind and Dr. I. S. Evans of the Census Research Unit, Department of Geography, and Dr. P. G. Barker of the Computer Unit, also at Durham, for their comments on the paper—but accepts full responsibility for the views expressed.

REFERENCES

1. M. Visvalingam, 'Storage of the 1971 UK census data—some technical considerations', *Working Paper 4, Census Research Unit*, Department of Geography, University of Durham (1975).
2. M. Visvalingam and D. W. Rhind, '*Compaction of the 1971 UK Census Data*', to be published.
3. D. W. Rhind, 'Geographical analysis and mapping of the 1971 UK census data', *Working Paper 3, Census Research Unit*, Department of Geography, University of Durham (1975).
4. D. Lefkovitz, *File Structures for On-line Systems*, Spartan Books, New York, 1969.
5. U. W. Pooch and N. Nieder, 'A survey of indexing techniques for sparse matrices', *Comput. Surv.* **5**, No. 2, 109–133 (1973).
6. J. K. Reid, 'Direct methods for sparse matrices', in *Software for Numerical Mathematics* (Ed. D. J. Evans), Academic Press, London, 1974.
7. A. Rosenfeld, *Picture Processing by Computer*, Academic Press, New York, 1969.
8. E. L. Amidon and G. S. Akin, 'Algorithmic selection of best method for compressing map data strings', *Comm. ACM*, **14**, No. 2, 769–774 (1971).
9. R. A. Wagner, 'Common phrases and minimum space text storage', *Comm. ACM*, **16**, No. 3, 148–152 (1973).
10. A. Mayne and E. B. James, 'Information compression by factorising common strings', *Comput. J.* **18**, No. 12, 157–160 (1975).