



Title	冗長度の低い符号を用いたバイナリ・データの縮約
Author(s)	浜辺, 賢一; 藤田, 勝美; 加地, 郁夫
Citation	北海道大學工學部研究報告, 84, 67-73
Issue Date	1977-07-11
Doc URL	http://hdl.handle.net/2115/41418
Type	bulletin (article)
File Information	84_67-74.pdf



[Instructions for use](#)

冗長度の低い符号を用いたバイナリ・データの縮約

浜辺賢一* 藤田勝美* 加地郁夫*
(昭和51年12月27日受理)

A Compacting Program for Binary Data File with low Redundancy Codes

Ken-ichi HAMABE, Katsumi FUJITA and Ikuo KAI
(Received December 27, 1976)

Abstract

A high-redundancy binary file, in which most of the bits are '0', is transformed into a more compact one by means of minimum-redundancy or optimum codes.

9 pairs of transforming (encoding) and decoding programs are made. Each pair has different codes matching to different bit-pattern of data. A property of the bit-pattern of a data-block and a corresponding program to be selected are determined by a bit-1 probability of the block.

The usefulness of an optimum coding regarding each block is also surveyed by the bit-1 probability of the block, and if it seems of no use, such a block is straightly copied without optimizing.

A user of the program can set an upper limit for a relative final size of the optimized file. If this limit is exceeded during an execution of encoding, that execution is soon aborted.

The original data file is recovered from the optimized data file by running it through a decoding program.

An actual performance of the program is presented in this paper regarding a 13-block file on disc-pack, which has different bit-1 probabilities from 4% to 56% for each block. After optimum-encoding with FANO-codes, the length of the optimized data file is shortened into 79% of the original data file.

By decoding of the optimized file and by a cross-reference with the original file, the validity of the whole program is verified.

It requires about 0.11 seconds of CPU-time for 1000 words of the original data, including both encoding and decoding.

1. ま え が き

C. E. Shannon を初めとする人々によって発展させられた情報理論 (Information theory) によれば, ある内容を持った情報 (例えば文章) は, いくつかの通報 (Message) (例えば文字) によって構成され, 通報はさらに符号 (Code) に変換されて機械的に取り扱うことが可能になるとみなされる。一般に符号化 (Coding) とは, 原情報の通報列と, 符号列との一対一の対応づけの過程といえるが, その中でも, 原情報内の同一情報量が, 可能な限り短い符号列に対応するような符

* 電気工学科 系統工学講座

号化の方法を、最小冗長度符号化 (minimum redundancy coding) あるいは最適符号化 (optimum coding) という。本研究は、このような符号の一つである FANÖ 符号を用いて、記憶装置上のファイルを縮約するプログラムを開発すると共に、理論との比較を行ったものである。

2. 基礎的事項

2.1 情報量の定義

Shanon によれば、一般に確率 $p_1, p_2, p_3, \dots, p_n$: $\sum_{i=1}^n p_i = 1$ なる n 個の事象のうちの 1 個が起る場合、これによって得られる平均 (期待) 情報量は次式で与えられる。

$$I_0 = -\sum_{i=1}^n p_i \log_2 p_i \quad (\text{ビット}) \quad (1)$$

(1) 式の値はまた、確率 p_i をそれぞれ持つ n 個の通報 u_i ($i=1, 2, \dots, n$) を二元符号化した際に得られる平均符号長の下限值である。実際には、連続した情報源記号を通報に区切るとき、1 通報の長さを無限に長くしなければ (1) 式の値は一般的には実現できない。

2.2 実用的な低冗長度符号— Shanon, Fano および Huffman 符号

以下はすべて二元符号とする。また、各通報 u_i は、出現確率の高い順に並べられているものとする。すなわち

$$p_1 \geq p_2 \geq p_3 \geq \dots \geq p_i \geq \dots \geq p_n \quad (2)$$

Shanon の符号は、通報 u_s に対する符号として、次式の累積確率、

$$P_s = \sum_{i=1}^{s-1} p_i \quad (3)$$

で表わされる P_s (二進小数) の小数部を、対応する原確率 p_i の小数点以下に初めて 1 が出現する桁数と同じ桁数だけとったものである。

Fano の符号は、(2) のようにならべた各通報を、それぞれの群の確率の累積和ができるだけ等しくなるような前後の 2 群に分け、それぞれの群に属する通報の符号に先頭ビットとして 0 と 1 を与える。次に各々の群をまた前後の 2 群に分け、符号の第 2 ビットとして 0 と 1 を与える。以上を 2 つ以上の通報をもつ群がなくなるまでくり返して得られる。

Huffman の符号は、まず (2) のようにならべた通報群のうち、最も出現確率の小さい通報 (u_{n-1} と u_n) の符号の末尾のビットとしてそれぞれ 0 と 1 を与える。次に u_{n-1} と u_n を統合して両者の確率 (p_{n-1} と p_n) の和をもった新しい仮の通報を作り、改めて $n-1$ 個の通報について再び出現確率の最も小さい 2 個の通報 (又は仮の通報に統合された原通報) の、これまでできている部分符号の先頭に 0 と 1 を加える。以上のことを全体が確率 1 の 1 通報に統合されるまでくり返すことで得られる。

Shanon 符号および Fano 符号は、通報の数が無限に多いときだけ最適な符号化を与えるが、Huffman 符号は通報数が有限の場合でも最適である。

また、これらの符号で、ある通報群を符号化し、相異なる 2 通報に対応する符号をその先頭のビットをそろえて並べた場合、一方の符号が他方の符号の一部と一致することは決してない。従って多数の通報の連続した情報を符号化した場合、通報の区切りに特別な符号は必要とせず、復号側では連続した符号を順次もとの通報に復元できる。このような符号を瞬時復号可能 (instantaneously decodable) な符号であるという。

いずれの符号を用いた場合でも、確率の高い通報には短い符号が、低い通報には長い符号が対応する。

図1に簡単な例で上の3つの符号化の実際を示す。

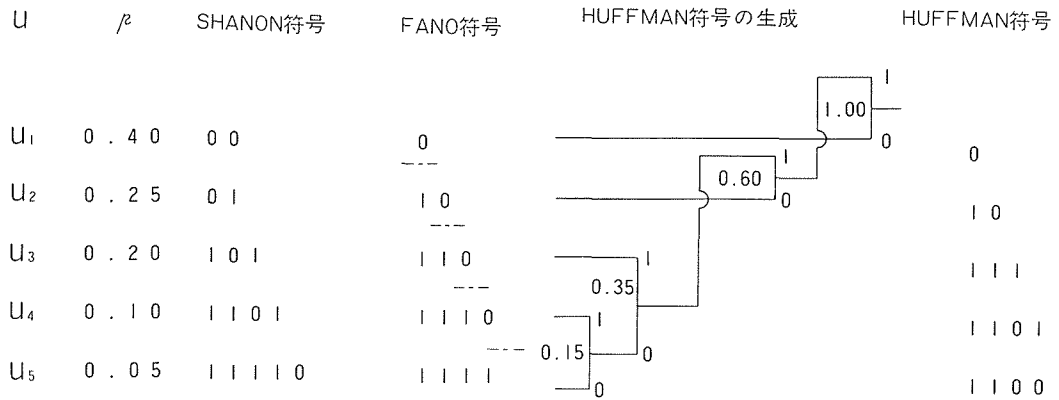


図1 5個の通報 ($u_1 \sim u_5$) に対する SHANON, FANO および HUFFMAN 符号の実際

2.3 バイナリ・データの最適符号化

このプログラムでは、符号化の対象となる原情報は計算機の内部、あるいは外部の記憶装置上にある情報であり、具体的には0または1のバイナリ (2値) データの連続したものである。また、符号化の単位となる通報としては、データを4ビットずつ区切ったものを用いる。いま、各々のビットは一定のビット1の出現確率 q を持ち、1の出現について相互の関連はないものとする、1のビットを m 個含む4ビット通報が出現する確率 Q_m は、各通報について

$$Q_m = q^m(1-q)^{4-m} \tag{4}$$

と表わすことができる。こうしてすべてのビットパターンに対する出現確率が知られば、前節にのべたような方法で各種の最適符号を与えることができ、これらの符号を用いてある領域 (語数) を占めるデータ全体もまた、最適符号化された別のビットパターンをもつデータに変換される。

前節に記したように、ここで用いる符号は、本質的には多数の連続した通報を符号化する際にも符号以外の情報は不必要であるが、固定長記憶単位 (語) をもつ計算機内部の処理では、各符号の符号長 (ビット) をパラメータとして併用するのが便利であり、これによって例えば、(110) 又は (00110) がレジスタ上に存在するときの相異を効率よく認識するのに役立つので、本プログラムでもそのようにしている。

3. プログラムの構成

計算機の記憶装置上のビットパターンを最適符号化しようとする場合、最も効率の良い符号の選び方は、そのデータの0, 1パターンのテクスチュア (“地合”) に依存する。それは、データのテクスチュアが、各通報パターン (この場合は4ビット長) の出現確率を決定するからである。データと符号との適合性を保ち、かつ高速の処理を可能にするために、このプログラムでは各ビットは0, 1の出現について独立という仮定の下に (これは1次のテクスチュアのみを考慮したことになる)、4%から36%までの4%ずつ増加する9種のビット1の出現率をもつ原データを予想し、これらのデータを4ビット長を1通報として最適符号化するための9種類の最適符号化ルーチンおよび最適化データを原データに戻すための9種の復号化ルーチンを用意した。

実際の使用に当たっては、原データ・ファイルの各ブロックから、ランダムな間隔をもつ100語

(データが1ブロック100語に満たないときはその全語数)をサンプリングし、各語の第0ビットから3ビット間隔で12ビットをしらべ、その結果求めたビット1の出現率に最も適合するルーチンを使用する。

1 通報の長さを4ビットとしたのは次の各条件を考え合わせた結果である。

- i) 1通報長は計算機で用いる1語=36ビットの約数であるのが望ましい。
- ii) 前記の如く、同一データを異なる1通報長による最適符号を用いて符号化した場合、データの短縮効率は、1通報長が長いほど良い。
- iii) 1通報長が長くなると、出現率の小さい通報ビット・パターンに対応する最適符号の長さが36ビット(1語長)を越え、プログラムがきわめて複雑になる。

1通報を6ビットとすると上のiii)の現象が起ることがある。

4. プログラムのアルゴリズム

4.1 このプログラムの対象となるファイル

実際のプログラムでは、処理の対象となるデータは外部記憶装置上のデータ(ファイル)とする。これは、もしデータの総量が計算機の主記憶装置上に存在しうる程度ならば、特にこのようなプログラムを用いてデータの量を縮小する必要性は少いからである。

ファイルに関するパラメータはカードによって与える。また、復号化の際のファイル・アクセス命令の中で必要なファイルの各ブロック長などのパラメータを記録するために、大記憶上に5トラックの補助ファイルを準備する。

カードから与えるパラメータは次の通りである。

- i) 最適符号化プログラムの場合：入・出力ファイルのファイル定義名、装置名、入力ファイルのブロック数、1ブロックの長さ(バイト)、入力ファイルのレコード型式、出力ファイルの要求短縮度、補助ファイルのファイル定義名。
- ii) 復号化プログラムの場合：入力ファイルのファイル定義名、装置名、補助ファイルのファイル定義名、出力ファイル作成の要否判定パラメータ、(必要ならば)出力ファイルのファイル定義名、装置名。

また、現在のところ、入力(原データ)ファイルについて次のような制約がある。

- i) 入力ファイルは固定長のブロックのみから成るファイルとする。
- ii) ブロックの総数は5600個以下とする。
- iii) 1ブロックの長さは32749バイト以下とする。

4.2 符号化プログラムのアルゴリズム

- 1) カード上のファイルパラメータの読み込み。
- 2) 入力(原データ)ファイルより1ブロックの読み込み。
- 3) 当ブロックの最適符号化の実行の可否判定。——このブロックのビット1の出現率(q)を、3.でのべたサンプリング測定し、 q の値が、最適符号化を行った際に、原データの長さとお適データの長さがほぼ等しくなるような値——このプログラムでは $q=0.4$ とする——よりも小さければ、最適符号化を実行する価値があるとみなして4)を実行し、 $q \geq 0.4$ ならば最適符号化は無益とみなして5)へ進む。
- 4) q の値に最も適合する最適符号化ルーチンによってこの1ブロック分の原データを最適符号化し、出力(最適化データ)ファイルの1ブロックとして書き込む。この際、ブロックの先頭の1語には、符号化に用いたルーチンの番号が、第2語には原データの1ブロッ

クの話数が書き込まれる。これは復号化に必要なパラメータの値を与えるためである。また、最適符号化データ・ブロックの長さでブロック番号を主記憶上に一時的に記憶する。その後 6) へ進む。

- 5) 原データブロックをそのまま最適符号化（出力）ファイルの1ブロックとしてコピーし、そのブロック長として、原データの1ブロック長（バイト）に 10^3 を加えた値を主記憶上に一時的に記憶する。その後 6) へ進む。
- 6) このブロックの最適ファイル化の終了時まで、原データ的全ブロック数の $1/5$ 以上の処理が終っており、かつこのブロックまでの最適化データ・ファイルの累計長が、対応する処理済みの原データ・ブロックの累計に対し、あらかじめ入力パラメータとして要求した限界値を越えたならば、それを当ブロックの番号と共にライン・プリンタに出力し、プログラムの実行を直ちに中止する。最適化ファイルの累計長が限界値内ならば、7) へ進む。
- 7) 入力ファイルの全ブロックの処理が終れば、出力（最適化）ファイルの各ブロック長その他のパラメータの値を、補助ファイルに書き込み、8) へ進む。末処理のブロックが残っていれば、2) へ戻る。
- 8) 最適化データ・ファイルの総バイト数、およびその入力ファイル総バイト数との比率、最適化データ・ファイルの各ブロック長の値をライン・プリンタに出力する。

4.3 復号化プログラムのアルゴリズム、及び両プログラムの内部処理の大略

- 1) カード上のファイルパラメータの読み込み。
- 2) 補助ファイルの内容の読み込み。
- 3) 入力（最適化）データ・ファイルの1ブロックの読み込み。
- 4) 復号化の要否の判定——補助ファイルから読み込んだ最適化ファイルのそのブロックの長さ（バイト）を表わす数値が 10^3 以上ならば、復号化は行わずに 6) へ進む。ブロック長が 10^6 未満ならば 5) を実行する。
- 5) 復号化ルーチンを実行し、復号化データの1ブロックを主記憶上に作る。実行すべきルーチンは、そのブロックの最適符号化の際に使用された符号化プログラムの中のルーチンと同じ番号のものであり、最適化データブロックの先頭の1語の内容によって自動的に選択される。その後 7) へ。
- 6) 最適化データ・ファイルの1ブロックをそのまま復号化データの1ブロックとして主記憶上にコピーする。その後 7) へ進む。
- 7) 必要ならば（カード・パラメータによる指定があれば）5) または 6) によって作成された復号化データの1ブロックを、改めて別のファイルに出力する。
- 8) 全ブロック終了まで 3) 以下を反復する。

実際の符号化・復号化の処理はアセンブラ・プログラムによって行う。その中心となる部分は次のようなものである。

FACOM-230-75 には、A レジスタ、R レジスタと呼ばれる各々 36 ビットのレジスタがあり、A レジスタを左方（高位）に、R レジスタを右方に置いて、連結した 72 ビットのレジスタとしてシフト演算を行うことができる。そこで、いま入力データ（符号化の場合なら原データ、復号化の場合なら最適符号化データ）の1語分を R レジスタに置き、AR レジスタの連結シフトによって、符号化の場合ならば 4 ビットづつ（復号化の場合ならば 1 ビットづつ）、左方の A レジスタにシフトし、A レジスタの内容に対応する出力データすなわち、符号化の場合は最適符号、復号化の場合は（もし存在すれば）原データの 4 ビット通報パターンをそれまでに作成さ

れた出力データの末尾に OR 演算によって接続することをくり返せば、出力データが得られる。

5. 実際のプログラム使用例

このプログラムを、大記憶装置上にある 2000 語×13 ブロックのバイナリ・データファイルについて実際に符号化・復号化の実行をさせた。

原データは、FORTRAN 基本外部関数である RANDOM (0) を利用して、ブロックごとに一定の出現率でビット 1 をランダムに分布させたものであり、その想定したビット 1 の出現率は、第 1 ブロックの 4% から 1 ブロック毎に 4% ずつ増加し、第 13 ブロックのみ 56% である。(実際のビット 1 の出現率は想定値とは僅かの差がある)

このファイル を Fano 符号化した結果として、ファイル全体として 79% の長さに短縮された最適化データ・ファイルが作成された。各ブロックの実際のビット 1 の出現率とブロック長の短縮度との関係は図 2 に示す。また図 2 には、式 1) の関係から算出される原データ 1 ビット当りの正味の情報量——理想的な最適化が行われた場合の短縮度——を共に示す。最もビット 1 の出

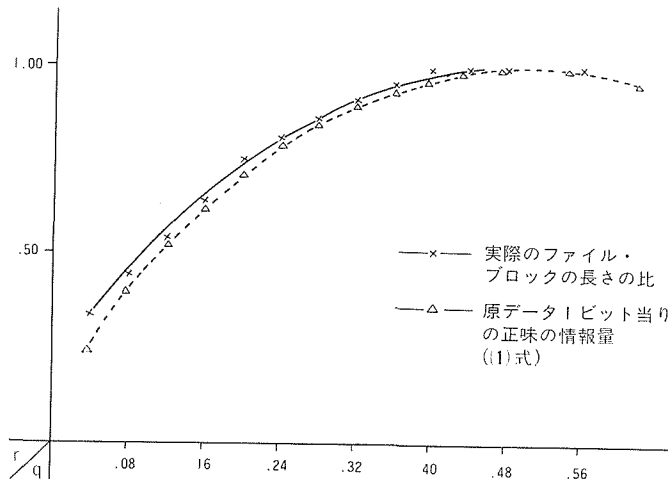


図 2 最適化データの各ブロックの長さ(バイト)の原データ・ブロックの長さに対する比 (r) と、ビット 1 の出現確率 (q) との関係

```

RAW DATA      BLOCK#  4
100000101010000000010000101000000100 000000000000101001000100010010000010 00000010000000000000100100000000000
00000000011001000100000010100001000 001000010000000001001011011001110000 00010000100000000001100101000011100
10000001000000000001000010010000000 00010000000000010000100000100010000 0000000000000100010000001000000100
00000001000001000010000100001010001 000000001000001011110000000100000 000001000000100000100001001001000000
00100000000100000000010001100000000 00010000100000000000000001000000000 0000011010001000000001000001010110000
00000000010000010000010000000000110 0000000010000100000000000000000000 000000101001000010100000100001000000
0000100010000000100000000000000000 10000001101100000100000010010010101 0000000100001000000000000000100010000
100100000110000000100010010110000000 000000101000000000000001100011000000 001000000000000100001000000000000100
0010001000000010010001000000000001 000000000000000001001001011000000000 000000000000010000100001011100000001
0000000000000010100000110000000000 0000000000101000000000000000100000 00000000000000000001010000010010000

```

```

OPTIMUL DATA  BLOCK#  4
00000000000000000000000000000000100 0000000000000000000000001111010000 1100101011011010001101101011000111
01101101101101110010100100000011010 00000111010101010011010110010101000 01011111110011100111001111010100011000
01111011011011100111101100100001100 101110001000001001011001000000101 11011010101010100110010010000100010
1100001011001111011110010100010 100101110010101110001010001100010 010010001000110000010100011001100110
11000100101011110000010110001011 001110000110010100000101011010011 1011010010110011001000101100001100
1001111100010100101111010110101000 11000010010001110100111000101010101 0111000010101000010011001111001010
00100011000010111010101001001010111 00010000001101010101111000001011 0101001111101010000001101010111100
00010101100000101000000100101011011 0010101110100111010110011110001011 0110011110111100010101010101000101
11010101010001010110001010000111 1101010111000101111010010100001000 101010001000110100111110000011010
000001001000110000001100001100010100 11000010110111011010001000011000011 01101001000000101011100101001000010

```

図 3 実際の原データと最適化データのブロック初頭部 (バイナリ表示) q=0.1558

現率の低い第1ブロックは原データの34%の長さに短縮されている。末尾の4ブロックはビット1の出現率が高く、最適符号化の利益がないか、或いはない可能性が高いので、原データのまゝ最適化ファイルにコピーされている。

原データ・ブロックと最適化データ・ブロックの先頭の部分を、実際のビット・パターンで図3に示す。

こうして作成された最適化ファイルを復号化し、原データ・ファイルとの照合試験を全ビットについて行い、完全に一致することを確かめた。

プログラムの実行必要時間 (CPU 時間) は、大記憶からの入・出力に要する時間も含め、符号化について原データ 1000 語当り約 42 ミリ秒、復号化につき約 69 ミリ秒である。両者を合わせた実行時間 1 秒当りの原データ換算処理可能語数は約 9000 語弱となる。

謝 辞

本報告のまとめに当って貴重な示唆を頂いた系統工学講座助手・二階堂正直氏に感謝の意を表す。

本研究は、北大大型計算機センターの昭和 51 年度ライブラリ・プログラム開発課題の一つとして、同センターの FACOM-230-75 を使用して行われた。

参 考 文 献

- 1) C. E. Shannon. 長谷川・井上訳: コミュニケーションの数学的理論 (1971), 明治図書.
- 2) 滝 保夫・宮川 洋: 岩波講座, 基礎工学 19, 情報論 I (1973), 第 3 章, 岩波書店.
- 3) 関 英男: 情報理論 (1972), 第 4 章, オーム社.
- 4) 笠原芳郎: 情報理論と通信方式 (1965), 第 3 章, 共立出版.
- 5) 富士通: FACOM 230-60 FASP 解説編 (1969).
- 6) 前谷・上窪・貝田・長田: ファイル・アクセスのためのサブルーチンについて(1), 北大大型計算機センターニュース・サプplement No. 8 (1975).