N91-24068

Recent Advances in Coding Theory for Near Error-Free Communications

K-M. Cheung, L.J. Deutsch, S.J. Dolinar, R.J. McEliece, F. Pollara, M. Shahshahani, L. Swanson

Communications Systems Research Section – Jet Propulsion Laboratory [*]

1 Introduction

In 1948 Claude Shannon revolutionized communication technology forever with the publication of his classic paper "A mathematical theory of communication." In this paper he wrote

> "The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point."

He then showed that the most efficient solution to this fundamental problem necessarily involved *coding* the selected message, before transmission over the channel. Shannon did not say exactly how this coding should be done; he only proved mathematically that efficient coding schemes must exist. Since 1948, a whole generation of later researchers has validated Shannon's work by devising explicit and practical coding schemes, which are now part of practically every modern digital communication system. Near-earth satellite communication systems, high performance military systems, computer communication networks, high speed modems, and compact-disk recording and playback systems, all rely heavily on sophisticated coding schemes to enhance their performance.

The communication channel which has benefitted most from Shannon's insights, perhaps, is the *deep-space channel*, i.e., the channel by which spacecraft like *Pioneer*,*Mariner*, *Voyager*, etc., transmit their astonishing views of the Solar System back to planet Earth. In this paper we will discuss the benefits that coding has already brought to deep-space communication, and the benefits that may yet become a reality.

Coding has traditionally been divided into two branches: *channel coding* and *source coding*. The goal of channel coding is to protect the transmitted message from the errors and distortions which may be caused by the channel. The goal of source coding (sometimes also called *data compression*), on the other hand, is to ensure that the transmitted message is as dense with information as possible. While channel coding has historically been the more important contributor to deep-space communication technology, future gains may come largely from source coding.

2 Channel Coding

Space communication engineers realized, almost from the beginning of the space age, that Shannon's ideas could be used to improve the design of spacecraft telemetry systems. Scarcely 10 years after the launch of the primitive *Explorer* spacecraft in 1958, NASA had begun the routine use of channel coding to enhance deep-space communications. In 1968, the *Pioneer 9* solar orbiter was launched with an encoder for a rate 1/2, constraint length 25 convolutional code to be decoded by the Fano sequential decoding algorithm. Similar encoders and decoders were used on later *Pioneers* which

[*]This article represents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration.

explored Jupiter, Saturn, and Venus. In 1969, *Mariners VI and VII*, twin Mars flyby missions, were launched with a (32,6) biorthogonal code which enhanced their performance by 2.2 dB at a decoded bit error probability of 5×10^{-3} (all performance gains in this section are compared to uncoded data.) This biorthogonal code was also used on the 1971 *Mariner 71* Mars orbiter and the 1975 *Viking* Mars orbiter and lander. Then in 1977 with the launch of the *Voyager I and II* spacecraft, a new generation of deep-space channel coding began. *Voyager* used a rate $r = 1/2$, constraint length $K = 7$ convolutional code, which was decoded on the ground using Viterbi's revolutionary decoding algorithm. This system achieved a performance gain of 3.5 dB at a decoded bit error probability of 5×10^{-3}. The coding system on Voyager was further enhanced by an 8-bit (255, 223) Reed-Solomon code, which was used in concatenation with the convolutional code to produce the low bit error rates needed by the data compression scheme described in Sec. 3.1. This coding system, which has become a standard for NASA and its Deep Space Network (DSN), is also in use on the *Galileo*, *Magellan*, and *Ulysses* missions. However, there is more to the *Galileo* coding story, as we will see in the following section.

Other significant new results on efficient coding for severely bandlimited channels, as in the mobile satellite environment, have been developed at JPL [5], but cannot be described in detail in this article.

2.1 Large Constraint Length Convolutional Codes: The *Galileo* Code

As mentioned in the previous section, the $K = 7$, $r = 1/2$ convolutional code, enhanced at low decoded bit error probabilities by concatenation with a (255, 223) Reed-Solomon code, has been a NASA standard since the launch of *Voyager* in 1977. However, according to Shannon's theorems, the performance of this system could in principle be improved by a further 4dB at a decoded bit error probability of 5×10^{-3}. Motivated by this tantalizing fact, in 1982, DSN Advanced Systems undertook a long-term research effort to study advanced coding techniques which would yield significantly more coding gain than is available using the NASA standard codes. Since Shannon had shown that as one approached his theoretical limits, the implementational complexity would increase explosively, the specific target of this research was a 2dB improvement, about half of the maximum possible theoretical gain. The hope was that this research effort would yield improved coding systems for missions in the "far future."

The quest for the 2 dB coding gain took off in several directions from current codes. The research focused on the same basic concatenation of a Reed-Solomon outer code with a convolutional inner code, but the code parameters were allowed to vary to levels not feasible when the present NASA standards were developed. The research effort studied the effects of increasing the constraint length and decreasing the code rate of the convolutional code, and increasing the symbol size and optimizing the code rate of the Reed-Solomon code. Due to a higher predicted payoff in performance versus complexity, a significant advance in convolutional code parameters was attempted, whereas the Reed-Solomon code parameters were only varied slightly from those of the present Reed-Solomon code used on *Voyager* and *Galileo*.

In 1986 the quest was declared a success, when, after extensive computer searches, some codes were found which surpassed the 2 dB goal, the best code improving performance by 2.11 dB (See Fig. 1 for a comparison of several coding systems.) To achieve this gain, the convolutional code constraint length had been increased to $K = 15$ and the code rate decreased to $r = 1/6$, and a 10-bit (1023, 959) Reed-Solomon code was used as the outer code [11].

Then on January 28, 1986 the *Challenger* catastrophe caused a three-and-a-half year postponement of *Galileo*'s launch. During this delay, a search was conducted for an advanced convolutional code which could be used by *Galileo* as an experimental mission enhancement option. Luckily, the 2dB code search was by then complete, and its results were made available to the *Galileo* team. The actual $K = 15$, $R = 1/6$ code could not be used by *Galileo*, since the bandwidth of *Galileo*'s radio modulator limits code rate to be at least 1/4. However, a fairly simple modification of the search that lead to the "2dB" code led to the discovery of a $K = 15$, $r = 1/4$, which was adopted for use by the mission. Because of *Galileo*'s bandwidth constraints, this code's gain falls short of the full 2dB.

Still, the *Galileo* code, as it is now called, will realize between 1 and 2dB gain beyond the standard NASA code, which may significantly improve the science return for this important mission. When *Galileo* was finally launched in October of 1989, it carried with it an experimental encoder for the new code, which incidentally also required a mechanism for doubling the subcarrier frequency and symbol clock rate whenever the encoder is invoked. Thus did the results of a research effort aimed at the "far future" affect a mission in the immediate present! [6]

However, at launch time, no decoder for the experimental code existed. The design of the decoders for such codes, suitable for VLSI implementation, is a current research project at JPL. The evolution and the results of this effort are the subject of the discussion in the following section.

2.2 Decoder Design: The Big Viterbi Decoder

The higher coding gains and the ensuing higher information returns promised by new, powerful coding schemes are becoming a reality through the design and implementation of the complex decoders that are required to take full advantage of such codes.

In particular, the complexity of a Viterbi decoder depends on three main parameters: the constraint length K, the code rate r, and the information data rate. The major complexity driver is the constraint length, since the amount of hardware is increasing exponentially with K. A decoder for $K = 15$ has 2^{14} states and is approximately 256 times more complex than the decoder for the $K = 7$ Voyager code.

Such decoders cannot be implemented on a single VLSI chip, because of memory storage and throughput rate requirements. Therefore, one is forced to develop concurrent algorithms which can be used on a multiprocessor system. The Viterbi algorithm is inherently a parallel algorithm. However, a fully parallel implementation of a large Viterbi decoder requires an impractical amount of hardware. This leads to the fundamental question of how to efficiently exploit this parallelism and how to contain it into practical limits by introducing some sequential re-use of the available hardware, which inevitably reduces the overall decoding speed.

Extensive research performed at Caltech and at JPL on multi-processor computers consisting of a network of simple node-processors interconnected as a n-dimensional cube (Hypercube), suggested the possibility to implement the Viterbi algorithm on a hypercube computer. The Hypercube or n-dimensional cube is a natural topology for efficient implementation of the Fast Fourier Transform (FFT). The similarity between the Viterbi algorithm and the FFT, which can be described in terms of the same graph topology and different algebraic kernels, was exploited to show how a network of 2^n processors, interconnected as a n-dimensional cube with no shared memory, can be efficiently used to implement a Viterbi decoder with various degrees of parallelism. The interprocessor communication overhead can be reduced by using the traceback method [4], which completely avoids any survivor exchange or any global memory operation. This algorithm has been implemented and successfully tested on a 64-node general purpose Hypercube computer for $(K, 1/N)$ codes, with $K = 3, ..., 15$, with measured efficiencies as high as 65 % for $K = 15$.

After this first experience with parallel decoding algorithms, it was decided to develop a Viterbi decoder, capable of decoding convolutional codes with constraint length up to 15 for NASA's Deep Space Network. Prototypes of the Big Viterbi Decoder (BVD) are being built both in semi-custom VLSI and gate array technology, and will use 512 or 256 identical VLSI chips, respectively, distributed on 16 identical boards. The final implementation will probably use a single board populated by 64 VLSI chips. The projected data rate will be approximately 1 Mbit/sec which is well in excess of present mission requirements. Galileo, for example, sends telemetry data at a maximum rate of 134.4 Kbit/sec.

The decoder could be implemented using serial or parallel architectures, or with a hybrid serial-parallel approach. In a serial architecture, a single physical butterfly processor performs all 8192 butterflies, sequentially. In a hybrid approach, n physical butterflies are used, each sequencing through $8192/n$ butterflies.

A fully parallel architecture was chosen for the BVD, since it is more convenient, in terms of VLSI implementation, to sacrifice parallelism by using bit-serial arithmetic. Therefore, additions and comparisons of metrics are done bit-serially, and the results are sent serially to other butterflies.

The most challenging problem in the design of the decoder is the wiring of the chips. This wiring is based on a novel partitioning [3], [10] of the decoder's state transition diagram and it defines the new decoder's architecture.

When the constraint length K is large, it is desirable to take a modular, hierarchical approach to organizing the huge number of required elements. Many add-compare-select circuits can be implemented on a single VLSI chip, and many chips can be mounted on a single printed circuit board. The full decoder is implemented by wiring together the required number of chips and boards.

The connection diagram of the 2^{K-2} butterflies is a deBruijn graph [7]; the butterflies are nodes in the graph and the edges of the graph represent wires between butterflies. It is possible to split the set of butterflies into modules called boards and the boards into modules called chips, in such a way that a large proportion of the required connections between butterflies are implemented internally within the modules. The chips are all identical (see Fig. 2) and the boards are all identical. Furthermore, their internal structure does not depend on the size of decoder, and an appropriate number of these *universal* modules can be wired together to make any size decoder.

Consider as an example a $K = 15$ Viterbi decoder consisting of 16 boards and 512 chips. Each chip in this design contains 16 butterflies, and each board has 32 chips. However, the theory developed in [3] is completely general and produces a modular, hierarchical partitioning of any size deBruijn graph into any number of first-level and second-level subgraphs (boards and chips), according to an FFT-like connection pattern.

The total number of wires cannot be increased or reduced by any wiring scheme. However, it is advantageous to capture as many of these required connections as possible within identical, small, modular units (chips and boards). The board and chip modules defined by this FFT-like construction have the property that full Viterbi decoders of all sizes can be constructed by appropriately connecting identical copies of the universal module, without revising the internal wiring within any module.

The success of the Galileo coding experiment depends now on the implementation of the $K = 15$ Big Viterbi Decoder, which should be completed by late 1990.

3 Source Coding

Although the theory of source coding is almost as old as that of channel coding, applications to deep-space imaging have been much slower to develop. There are several reasons for this. First, it is quite difficult to produce credible mathematical models for image statistics — this is in sharp contrast to deep-space channel modeling, where the Gaussian model has served admirably for 30 years or more. Second, and perhaps more important, in source coding the burden of complexity is on the *encoders*, which must be on the spacecraft and so highly constrained in power, weight, etc., whereas in channel coding, the burden is on the *decoder*, which is on the ground and so relatively unconstrained. A further complication is the difficulty in formulating a meaningful distortion measure. Nevertheless, beginning with *Voyager's* 1986 encounter with Uranus, source coding has begun to play an important part in deep-space telecommunications systems.

3.1 Voyager's and Galileo's Data Compression Scheme

The *Voyager* mission, at Uranus in 1986 and Neptune in 1989, used a source coding technique pioneered by R. F. Rice, and achieved a compression ratio of 2.5:1. This algorithm is essentially a universal source code on the differences between successive pixels. An enhanced version of Rice's algorithm, known as the BARC (block adaptive rate controlled) scheme will be used on *Galileo's* images of Jupiter [9]. A general version of Rice's algorithm is now being developed for NASA in VLSI by the University of Idaho.

However, both of Rice's schemes (Voyager and BARC) were constrained by the need to use 1970's space-qualified hardware. With the availability of enormously more powerful VLSI hardware in the

1990s and beyond, it is now possible to envision a future in which source coding can deliver gains to deep-space telemetry fully comparable to those already realized by channel coding.

3.2 Current Research in Data Compression for Images

The theoretical limits of combined source-channel coding for a Gauss-Markov source and a Gaussian channel are illustrated in Fig. 3, where the performance is measured in terms of mean square reproduction error as a function of the signal-to-noise ratio of source symbols, for two values of bandwidth expansion factor $\eta = \infty$ and $\eta = 4$. The correlation coefficient γ represents the dependency between successive pixels. No communication system can be designed below the curves for $\eta = \infty$. For a practical bandwidth expansion factor ($\eta = 4$), it is shown how the exploitation of the spatial redundancy of a typical planetary image (with correlation coefficient $\gamma = 0.99$) offers a potential improvement of approximately 5 dB. These large potential gains motivate the current research in source coding.

In the area of lossless compression schemes, we have developed a method based on traditional DPCM (Differential Pulse Code Modulation) encoded with a GVH (Gallager-vanVoorhis-Huffman) code, which is a near optimal adaptive Huffman code using simple table look-up operations, and requiring very modest computation [1].

However, lossy compression schemes are the most promising for achieving large gains. The same GVH code has been applied to the coefficients resulting from a two-dimensional discrete cosine transform (DCT) on 8×8 or 16×16 blocks of pixels. This strategy yields a scheme with negligible distortion and high compression ratios (10:1 to 20:1 on typical images). This technique is based on the fact that the DCT coefficients are approximately geometrically distributed and the GVH code is optimal in this case.

For cases where higher distortion is allowable, a new locally adaptive vector quantization scheme has been developed with compression ratios in the range 30:1 to 60:1. This is a new class of source coding schemes based on simple dynamic codebook updating strategies, and on self-organizing data structures [2]. This scheme does not assume any a priori knowledge of the source statistics, nor does it require any preprocessing or codebook training. The codebook is generated on the fly, and is constantly updated to capture local features of the data.

4 Future Research Directions

4.1 Neural Networks for Soft Decoding

Block codes are traditionally decoded by first "hard quantizing" symbols, i.e. by replacing soft channel symbols by zeros and ones, followed by algebraic decoding. This has the disadvantage of losing about 2dB compared to soft, maximum-likelihood decoding. The problem of soft decoding of a (n, k) block code consists in finding the distances of a n-dimensional vector of soft channel symbols to the 2^k codewords, and in selecting the closest codeword. Unfortunately, known soft maximum-likelihood decoding methods are extremely complex, and so block codes have rarely been used as inner codes for deep-space applications, where the 2dB loss is too large.

Recent results obtained by JPL researchers show that neural networks can be efficiently used to implement such decoders, yielding highly parallel structures and extremely high decoding speeds. The decoding problem can be cast in a form suitable for multi-layer feed-forward neural networks (perceptrons). This can be accomplished by observing that the decision regions for the 2^k codewords are convex regions delimited by hyperplanes in n-dimensional space, and multi-layer networks can be configured to describe exactly such regions. Two layer networks are sufficient to solve this problem, while the conventional search for the minimum distance codeword implies $\log_2 2^k = k$ sequential stages or layers. Thus very fast decoding should be possible.

The number of neurons necessary for such decoders was analyzed together with the performance penalties incurred by reducing such number. The fault tolerance properties inherent in neural networks could be valuable in on-board or remote applications. Learning rules (the back propagation rule, etc.) allowing efficient use of the available neurons by finding the optimal weights to be assigned to the branches of the network are being developed.

The greatest potential of neural nets decoders is the high-speed processing that could be provided through massively parallel VLSI implementations.

4.2 Neural Networks for Source Coding

Recent research conducted at JPL has shown how a vector quantizer can be used for image compression by mapping a sequence of continuous or discrete vectors, representing a rectangular block of pixels, into a lower rate sequence, suitable for communication over bandwidth constrained digital channels.

Traditional vector quantizer encoders are computationally very intensive. Highly parallel feed-forward neural networks can be used to implement a class of memoryless vector quantizer encoders, with encoding time proportional to $\log_2 N$, in contrast to the KN of traditional encoders, where N is the codebook size and K is the dimension of each vector.

Feed-forward neural network implementations of a class of finite-state vector quantizer encoders are also being developed to take advantage of the correlation between successive source vectors.

It was estimated that these vector quantization schemes can provide a 10:1 compression ratio with little distortion, and a 100:1 compression ratio with moderate distortion for several sources. The squared error distortion measure is used for comparing different schemes, together with other subjective distortion measures.

Theoretical performance results were verified by simulation, and a preliminary design and architecture for VLSI implementation of the proposed compression systems is under development.

4.3 Finite-state Codes

Methods for obtaining large coding gains using a new class of hybrid trellis codes have been developed at JPL. These codes are constructed by combining block with convolutional codes in a novel manner, using the recent idea of "set partitioning" developed by Ungerboeck and others. We believe that these codes can significantly improve the performance of coded communication systems without requiring a significant increase in decoder cost.

A convolutional code is characterized by several integer parameters: k, the number of information bits that enter the encoder at each clock cycle; n, the number of transmitted bits that leave the encoder at each clock cycle; and m, the memory, which represents the number of previous k-bit information blocks that the current n-bit output block depends on. In principle, these parameters can assume any integer values, but for historical (and other) reasons almost all current implementations of convolutionally encoded systems use very small values for k and n, and rely on the encoder memory m to obtain the required coding gains. (For example, the convolutional code used by Voyager has $k = 1$, $n = 2$, and $m = 6$.) However, there are convincing theoretical arguments that show the existence of powerful and practical convolutional codes for which the parameters n and k are relatively large, while the memory m is relatively small. Unfortunately, it has always been very difficult to locate such codes because there is no really satisfactory algebraic description of them and brute-force computer searches are prohibitively difficult.

However, we have recently discovered a large class of convolutional codes with large n and k, and small m, which can be described algebraically, and which our preliminary studies have shown to contain some very powerful codes [8]. Our basic idea is as follows. We choose the parameters n and k to correspond to the block length and dimension, respectively, of a known block code. We then partition this block code into a number of smaller subcodes, and then assign codewords from these subcodes to the branches of the trellis diagram corresponding to a convolutional code with small memory, as shown in Fig. 4. The codes constructed this way can have higher coding gain than the original block code, increased ability to correct short bursts of errors, and less decoder complexity than traditional convolutional codes with the same level of performance. "Higher coding gain" will translate into lower decoded bit error probability; the short burst error correcting property will make these codes attractive as inner codes in concatenated coding systems, and the decreased complexity will make VLSI implementation feasible.

Additional advantages include the existence of higher rate codes (smaller bandwidth expansion) with good performance, and the possibility to avoid lengthy computer searches to find good codes.

4.4 Fractals for Data Compression

The use of fractals has been proposed to capture very complex images in a few parameters. This has the potential for huge compression ratios, at the expense of very complex encoding algorithms and moderate distortion.

The idea of the fractal method is to identify a data set with the stationary distribution of a Markov chain determined by a finite set of affine transformations. While a Markov chain can be specified by very few parameters, its stationary distribution may indeed be a very complex object. The goal is to take advantage of this fact and obtain concise representations of very complex objects or data sets by simply encoding the parameters of the Markov chain. Successful development of the technology for the implementation of this idea would have very significant implications for source coding.

While many images may be generated by the above mentioned method, the procedure requires human supervision and is very time consuming. Therefore the key issue is to systematize the procedure so that it can be implemented in an automated fashion. We have had some success with such a method for one dimensional data. In two dimensions, for instance for images, we face a new set of problems due to the complexity of the data. It has been suggested that by using the method of "skeletonization" one can determine the parameters of the Markov chain whose stationary distribution is the given image. We are presently experimenting with this idea.

References

[1] K. Cheung, P. Smyth and H. Wang, "A High-speed Distortionless Predictive Image-compression Scheme", TDA Prog. Rep. 42-102, JPL, Pasadena, Aug. 15, 1990.

[2] K. Cheung and V.K.Wei, "A Locally Adaptive Source Coding Scheme", submitted to IEEE Trans. on Communication Theory.

[3] O. Collins, F. Pollara, S. Dolinar and J. Statman, "Wiring Viterbi Decoders (Splitting DeBruijn Graphs", TDA Prog. Rep., JPL, Pasadena, Feb. 15, 1989, pp. 93-103.

[4] O. Collins and F. Pollara, "Memory Management in Traceback Viterbi Decoders", TDA Progress Report 42-99, Jet Propulsion Lab., Pasadena, CA, Nov. 1989, pp. 98-104.

[5] D. Divsalar and M.K. Simon, "The Design of Trellis Coded MPSK for Fading Channels", IEEE Trans. on Comm., Vol. 36, No. 9, pp. 1013-1021, Sept. 88.

[6] S. Dolinar, "A New Code for Galileo," TDA Prog. Rep. 42-93., May 15, 1988, pp. 83-96.

[7] S.W. Golomb, "Shift Register Sequences", California: Aegean Press, 1982.

[8] F. Pollara, R. McEliece, and K. Abdel-Ghaffar, "Finite-state Codes," IEEE Trans. Inform. Theory, vol. 34, No. 5, Sept. 1988.

[9] R.F. Rice and J. Lee, "Some Practical Universal Noiseless Coding Techniques, Part II", JPL Publ. 83-17, JPL, Pasadena, March 1, 1983.

[10] J. Statman, G. Zimmerman, F. Pollara and O. Collins, "A Long Constraint Length VLSI Viterbi Decoder for the DSN", TDA Prog. Rep. 42-95, JPL, Pasadena, Nov. 15, 1988, pp. 134-142.

[11] J.H. Yuen and Q.D. Vo, "In Search of a 2 dB Coding Gain", TDA Prog. Rep. 42-83, JPL, Nov. 85, pp. 26-33.

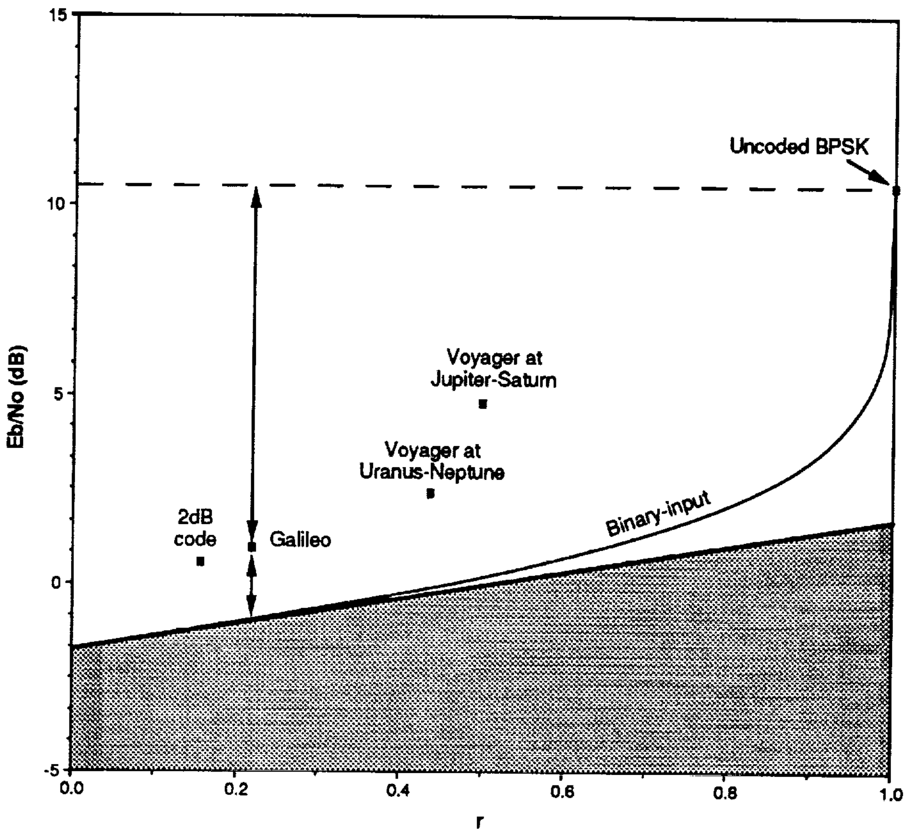


Figure 1 Performance comparison of several coding systems.

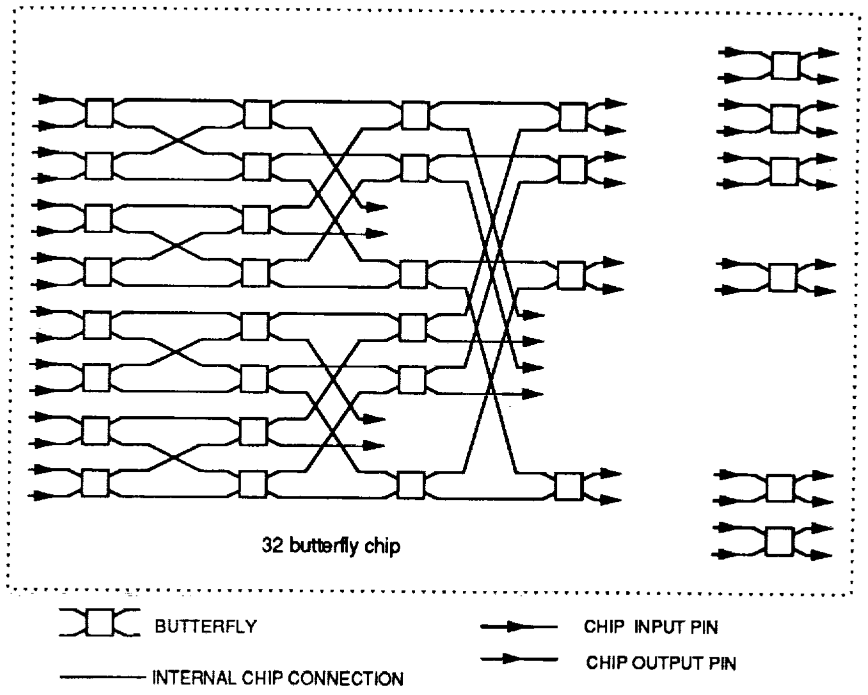


Figure 2 Example of universal BVD chip.

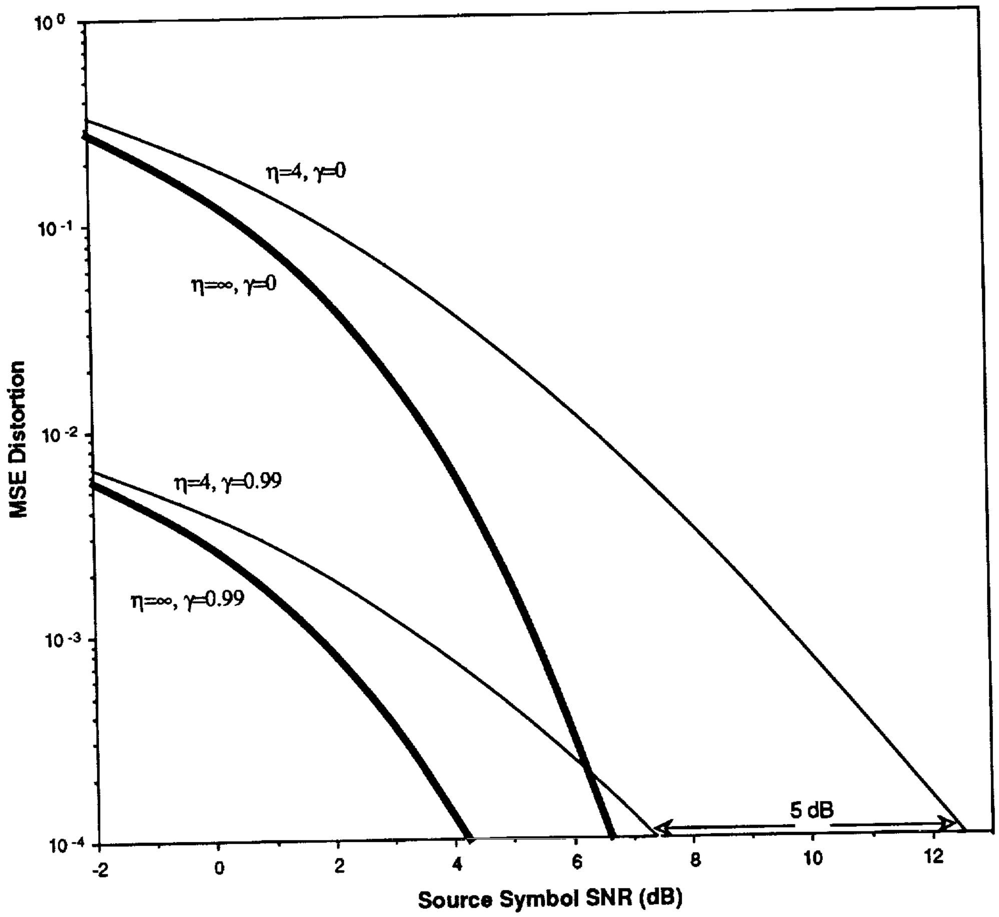


Figure 3 Performance gains of source coding

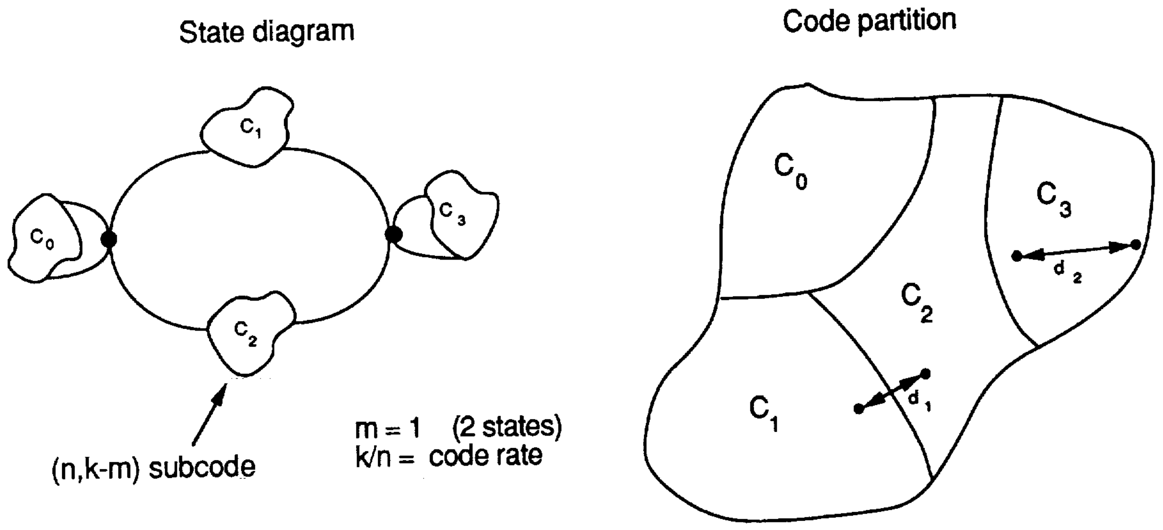
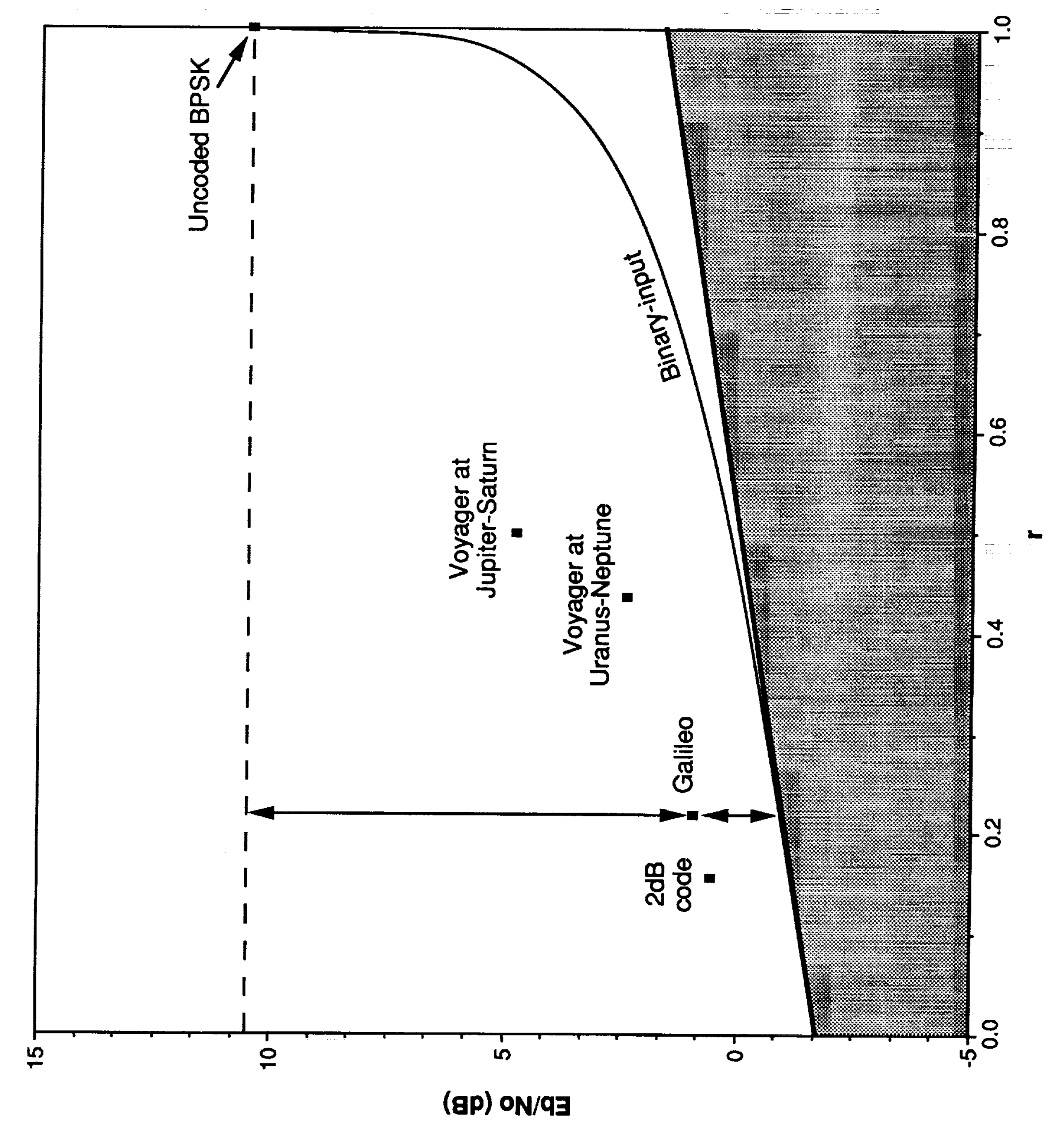
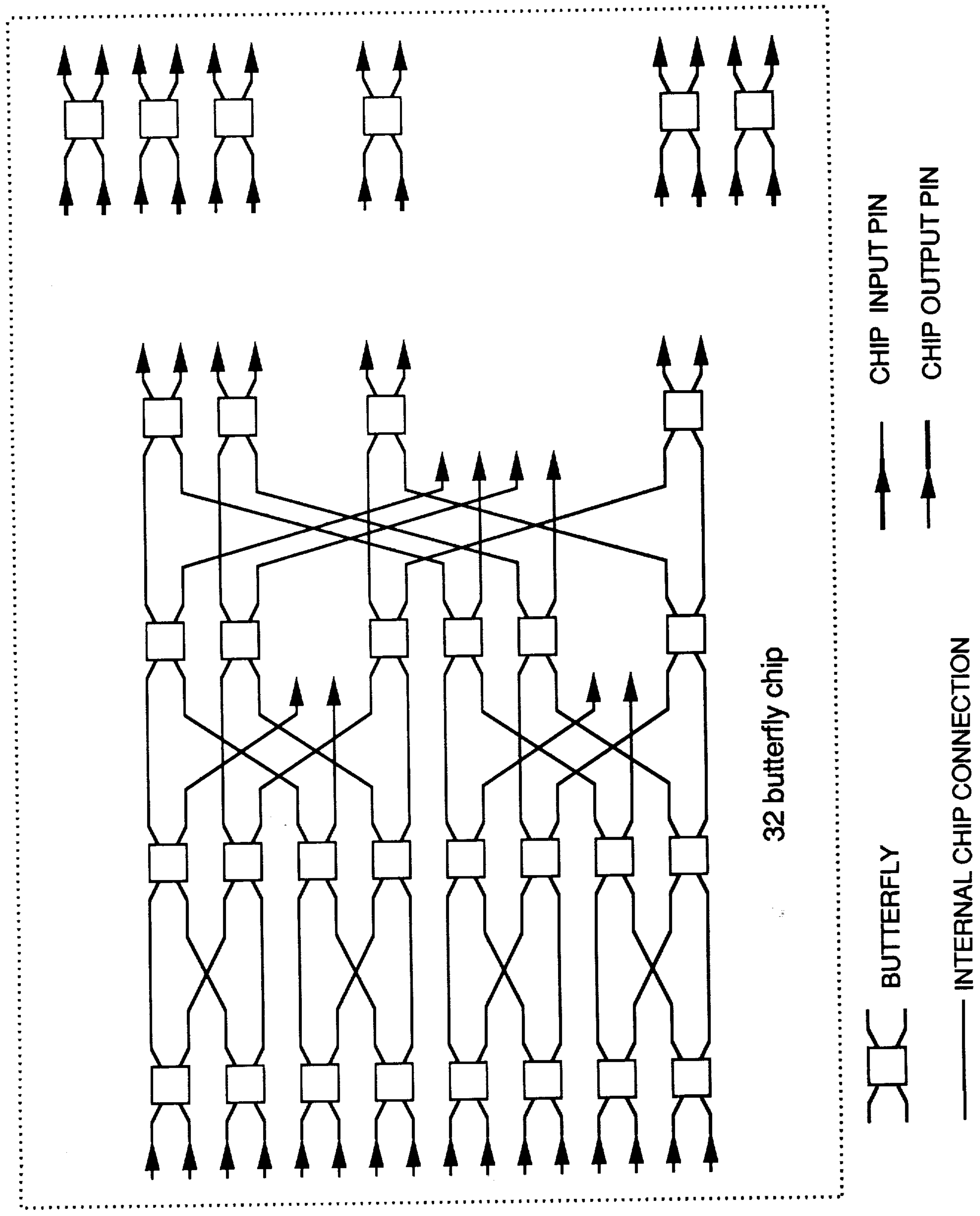
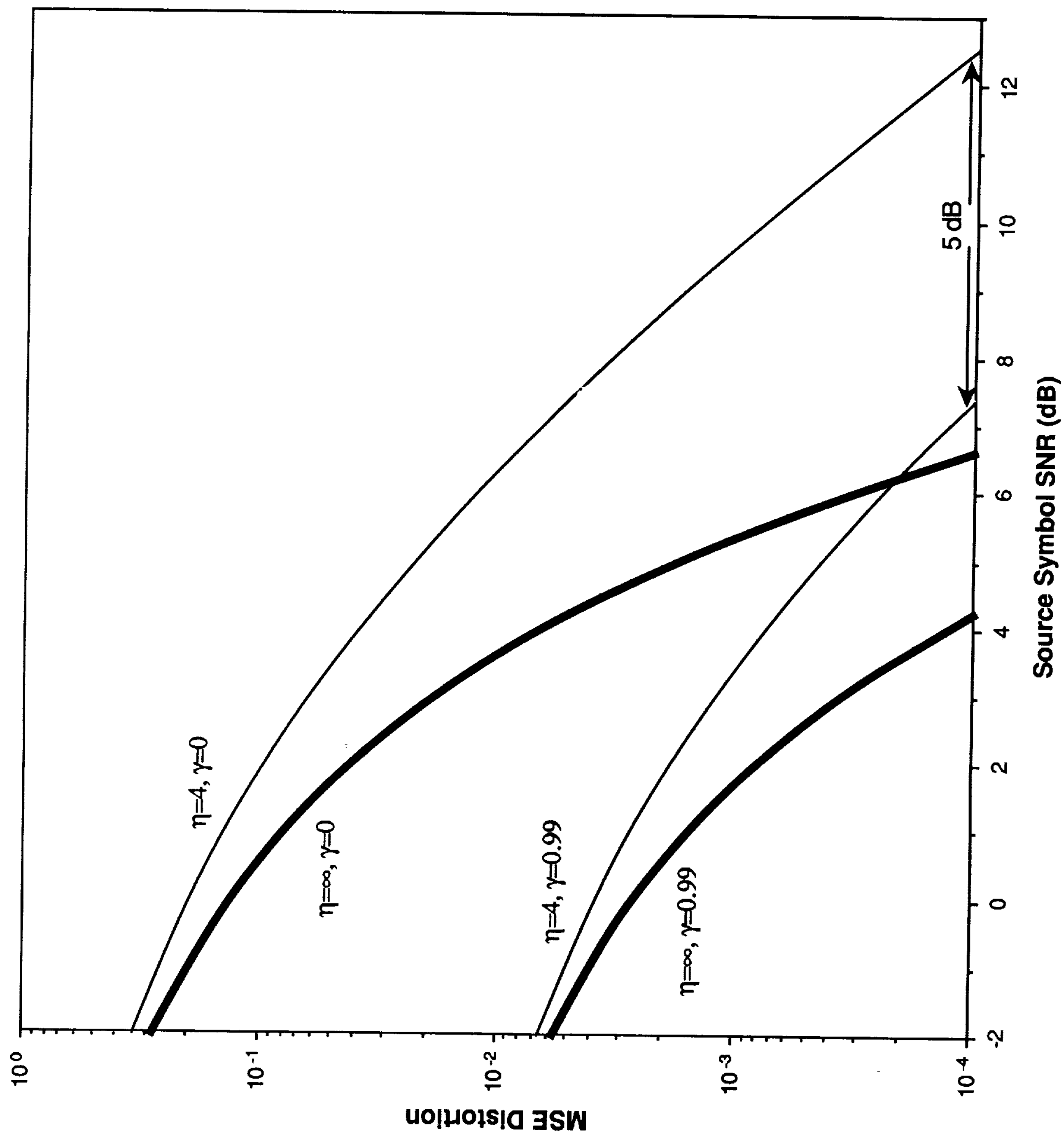


Figure 4 Example of finite-state code



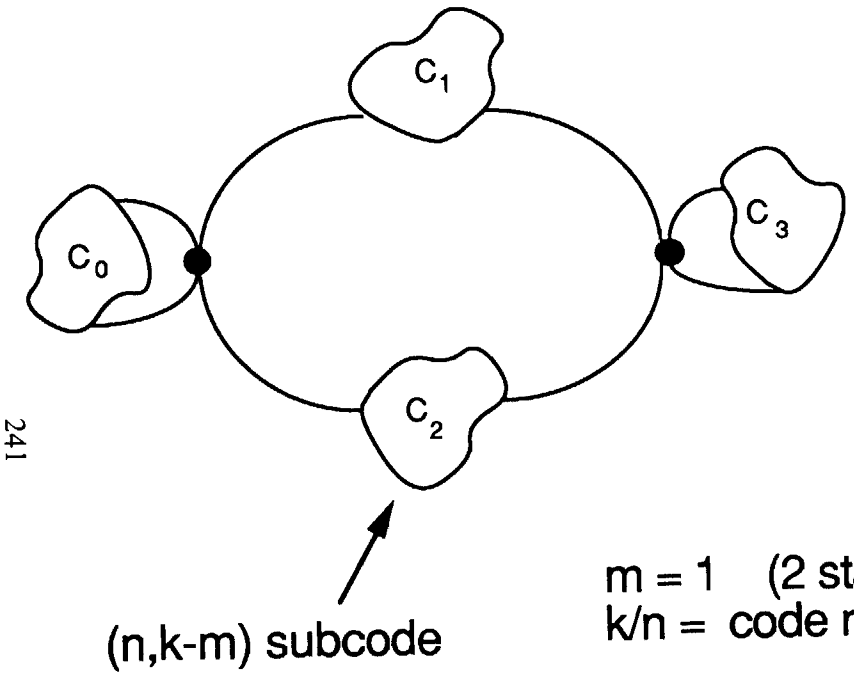
Eb/No (dB)
15
10
5
0
-5
0.0
0.2
0.4
0.6
0.8
1.0
r
Uncoded BPSK
Binary-input
Voyager at Jupiter-Saturn
Voyager at Uranus-Neptune
Galileo
2dB code


32 butterfly chip
BUTTERFLY
INTERNAL CHIP CONNECTION
CHIP INPUT PIN
CHIP OUTPUT PIN



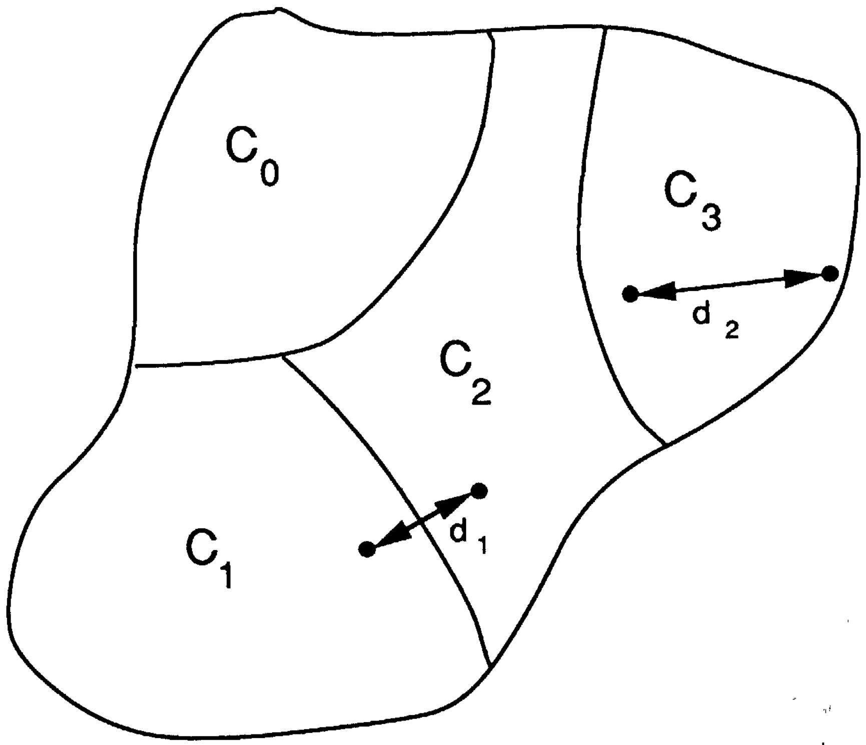
MSE Distortion
10 0
10 -1
10 -2
10 -3
10 -4
Source Symbol SNR (dB)
-2
0
2
4
6
8
10
12
η=4, γ=0
η=∞, γ=0
η=4, γ=0.99
η=∞, γ=0.99
5 dB

State diagram



m = 1   (2 states)
k/n = code rate

Code partition


C_0
C_3
d_2
C_2
C_1
d_1