算術符号とその画像符号化標準への適用

小野 文孝†

†東京工芸大学

E-mail: ono@image.t-kougei.ac.jp

あらまし 算術符号化はその対象系列ごとに独立で符号を決定できるという特性から，マルコフモデル符号化の実用的かつ効率的符号化手段として脚光を浴び，まず 2 値画像符号化標準に適用された．その後，適用範囲を拡大し多値画像符号化標準，動画符号化標準にも採用されている．これは算術符号がコンテクスト認知により汎用的なモデルに適用できるという効果に基づくものといえる．そこで本報告では算術符号化の各種標準への適用の歴史を振り返るとともに，その簡易化・高速化の歴史についても紹介する．最後に今後の算術符号化における検討項目についても展望する．

キーワード 算術符号，静止画符号化， ハフマン符号，符号化標準

Arithmetic Coder and its Adoption by the Image Coding Standards

Fumitaka ONO†

† Professor Emeritus, Tokyo Polytechnic University

E-mail: ono@image.t-kougei.ac.jp

Abstract In the arithmetic coding it is possible to allocate the codeword to the occurred symbol sequence independent to other possible sequences. Such feature makes it possible to encode Markov model source in practical and efficient way. Therefore arithmetic coding was firstly adopted in the Markov-model based bi-level image coding standard, and then it was adopted in still image coding standards and also in video coding standards. The reason of such expansion is that the arithmetic coding can realize the Markov model entropy rate through the automatically recognizing the context. In this paper, the author reviews the history of the adoption of the arithmetic coding by the image/video coding standards and also the history of the effort to speed up the processing way of arithmetic coding. The future prospect of arithmetic coding is also introduced.

Keyword arithmetic coder, still image coding, Huffman coding, coding standards

1. まえがき

算術符号の最初の実用的提案[1]は 2 値画像の符号化を目的とするものであった．その結果，算術符号の持つブロック符号（非算術符号）では成し得ないマルコフモデル情報源（マルチコンテクスト情報源）の情報源拡大符号化効果が評価され，マルコフモデルに基づく 2 値画像符号化の最初の標準である JBIG-1[2]におけるエントロピー符号手段としてまず採用された．

次に多値静止画標準への適用であるが，最初の多値静止画符号化標準である JPEG[3]では DCT（Discrete Cosine Transform）係数のランレングス符号化がモデルとして採用されたためハフマン符号で充分であったが，プログレッシブ符号化やロスレス符号化などへの適用も考慮して算術符号はオプション符号として採用された．続く，JPEG-LS[4,5]では多値マルコフモデルに基づく符号化となったため算術符号の採用が検討されたが最終的にはエントロピー符号／復号器の複雑性を考慮して必須機能である Part1[4]ではモード分離を行い自然画モードでは情報源拡大を行わないブロック符号が採用され，情報源拡大が必要な CG・文書画像モードではランレングス符号化が採用になり算術符号の採用は見送られた．その結果算術符号は JPEG-LS ではオプションである Part2 においてのみ採用されることになったが，Part2[5]ではモード分離を不要とすることができた．続く標準である JPEG 2000[6]では DWT（Discrete Wavelet Transform）係数をビットプレーン分解して符号化するモデルが採用され，ついに算術符号が唯一のエントロピー符号化方式に位置付けられた．

一方動画標準での算術符号の採用は H.264[7]が最初で CABAC がブロック符号である CAVLC と共存する形をとった．最新の H.265[8]では CABAC が唯一のエントロピー符号化となっている．

つまり算術符号は当初はマルチコンテクスト情報源の効率的符号化手段という機能が評価されて導入され，最近では符号化の絶対的な高効率性が評価されると同時に情報源のコンテクストの自動認知効果により達成目標エントロピーを低減化する効果も評価されるようになったといえる．

そのような観点から，算術符号の積年の課題である高速化や，これまで殆ど検討されていない多値算術符号についても，今後引き続き，より真摯に検討されるべきであろう．本文では算術符号の標準化への適用の歴史，簡易化・高速化の歴史を振り返るとともに今後の展開についても展望する．

2. ブロック符号によるマルコフ情報源符号化

算術符号の効果を考える上でブロック符号によるマルコフ情報源の情報源拡大を許したコンパクト符号化方式として「状態別拡大」符号化方式と「状態混在順次拡大」符号化方式について紹介する[9),10)].

2.1 状態別拡大方式

簡単のため 2 値の情報源を想定する．2 値の情報源では MELCODE[11)]のように，MPS（More Probable Symbol）のみを情報源拡大した汎用的・規則的な符号セットが知られている．N 次 MELCODE とは，MPS が N 個続いて出現するパターンに 1bit の符号語，それまでに LPS が出現する N 通りのパターンには符号長の差が最大 1bit の符号語を割り振るもので，LPS（Less Probable Symbol）の出現確率をパラメータとして最適な拡大次数 N が導かれ高い符号化効率が得られる．なお MELCODE はランレングス符号(LPS で終端)に適用すれば Golomb[12)]符号と符号割り当てが同じになるが，非ランレングスの情報源も対象とできるので MPS で終了する情報源への効率を Golomb 符号より高くできる．

今，コンテクスト（マルコフ状態）として A,B の 2 通りをもつ情報源があるとする．第 1 番目のシンボルのコンテクストが A であり，コンテクストごとの情報源拡大により，第 1 番目のシンボルを含むコンテクスト A の通報が完成するのが 10 番目のシンボルであったとしよう．そして，それまでにコンテクスト B の通報は例えば 2 度完成していたとする．受信側では，まず第 1 番目の出現シンボルを復号する必要があるので，最初に第 1 番目の出現シンボルを含むコンテクスト A の符号語が送信されなくてはならない．したがって，送信側では 10 番目のシンボルによりコンテクスト A の符号語が完成した時点でその符号語をまず送り，続いて既に完成済みのコンテクスト B の符号語 2 件をその発生順に送出する必要がある．

このように状態別拡大符号化ではコンテクスト毎の符号語が，通報の完成順ではなく，通報の先頭シンボルの発生が古い順に伝送されなければ，受信側での復号が不可能となる．したがって送信側では符号語の送信順序制御のための並べ替え作業(インターリーブ)と既に完成済みの符号語を蓄えておくメモリが必要になる．よって，状態別拡大符号化の符号設計と送信制御は以下のように記述される[9)].

［状態別拡大符号化方式］
状態別拡大符号化では，状態毎に，その符号化効率を基準に，必要となる通報ごとの拡大次数を求め，符号を設計する．符号語の送信は通報に含まれる先頭シンボルの発生が古い順に行うので状態間で並べ替えが必要になることもある．送信側の符号語メモリに上限があればメモリ制御も必要となる．　（以上）

このように「状態別拡大」ではインタリーブ処理が必要となること，メモリが有限であればそれがフルになった場合の制御（ダミーシンボルによる強制的な通報完成）も必要となることが欠点である．しかし，「状態別拡大」ではそれぞれの状態の出現シンボルのみをまとめて符号化でき，1 つの状態内では無記憶情報源出力となるため，2 値符号化のための確率パラメータは 1 つとなり，また，統計的性質が近いコンテクスト同士はまとめることができ，効率計算も容易化できるという利点がある．

2.2 状態混在順次拡大符号化

さて，ブロック符号によるマルコフ情報源の情報源拡大を許した符号化のもう一つの手段として，上に述べた「状態別拡大」の他に，開始状態毎に複数の符号群からその 1 つを選択する「状態混在順次拡大」符号化方式が考えられる[10)]．この方式は，「状態別拡大」符号化で必要とする符号のインタリーブが不要となるが符号の種類数が増加し設計もより複雑になる．

１） 符号設計問題

状態混在順次拡大の符号設計においては，開始状態においてシンボルのあらゆる出現形態を想定し，符号化効率を考慮して最大の拡大次数を設定して符号を構成する．これにより，開始状態ごとに適用する符号が求まることになる．これは情報源拡大も含めたハフマン符号の設計であり，たとえば情報源の拡大要件を MPS のみとし累積確率の上限を決めておくことで通報設定はできるとしても「状態分離拡大」におけるような系統的符号設計や評価は不可能であり，情報源に応じた個別の設計が必要になるといえる．

２） 符号種類数

次に，符号の種類数について考察するため，前節で導いた，「開始状態数」に着目する．まず，2 元 N 重マルコフ情報源においては，先立つ N 個の出現シンボルを知れば状態（状態数＝2^N）がわかり，それ以降の状態は復号結果により自動的に判明する．開始状態数はマルコフ状態数と同じであるので，順次拡大符号化における符号の種類数は，最大でマルコフ状態数と同じ 2^N 通りと考えればよい．

しかし，参照画素を直前に連続するわけではない N

個とするような場合は縮退マルコフ情報源であり，「状態別拡大」では必要となる符号の種類数が，縮退後の状態数 $R(=2^N)$と等しくてよいのに対し，「状態混在順次拡大」では，必ずしも R まで減少しない．

一般に，画像符号化では 2 次元相関の利用に際し，注目画素に近い位置の画素だけを参照する場合が多い．簡単な例として 2 値画像で注目画素の直上画素のみを参照して符号化する場合を考えると，この場合，状態は 2 通りに縮退されており，状態別拡大では，2 通りの符号（上が白の場合と黒の場合）を用意すればよいだけである．しかし，1 ラインの画素数を L とすると，この情報源は本来 L 重マルコフ情報源である．なぜなら，開始状態における縮退情報（直上画素の値）と，そのシンボルの復号結果からは続く画素の状態に関する縮退情報が得られないからである．とはいえ 2^L 状態まで分離する必要はなく，仮に情報源の 4 次の拡大までが必要とすると，注目画素の 3 画素先のコンテクストまでの組み合わせを考え 2^4 通りの開始状態について符号を用意すればよい．

したがって，開始状態とは，「情報源の最大拡大時における注目シンボルのコンテクストが特定できるための状態」といいかえられる．よって，状態混在順次拡大符号化の符号設計は以下のように記述される [10]．

［状態混在順次拡大符号化の符号設計］
順次拡大符号化では，開始状態毎に，その後の状態の遷移を調べ，符号化効率を基準に，最大拡大次数を求め，符号を設計する．その符号設計は情報源拡大時に起こりうるコンテクストの組み合わせに依存した複雑度をもつ．ここで開始状態とは，「最大拡大時の遷移におけるコンテクストが特定できるための状態」であり，この開始状態数に対応して符号を用意する必要がある．　（以上）

このように，状態混在順次拡大は，「状態別拡大」におけるインタリーブのような送信制御が不要という利点はあるものの，コンテクスト混在により符号設計がより複雑化すること，符号の種類数は状態の縮退が行える場合でも縮退状態数まで減少せず，情報源の最大拡大時の縮退状態までを特定できる開始状態の種類数に等しくなるという短所があることがわかる．

ただ，もし MPS 出現では状態が変化しないという捉え方をすれば，この場合は状態分離拡大と状態非分離順次拡大での符号化対象が一致するので，同じ符号が使えることになる．過去に，提案されている「スタートパターン別ランレングス符号化」[13]や，JPEG-LS Part1 の「ランモード符号化」[4]は，順次拡大の例といえるが，いずれも，縮退初期状態に応じて符号を用意し MPS 発生では同一状態の継続とみなして簡易化している．このような簡易化の結果，情報源によってはそれに伴い目標情報量も増大することになる [10]．

3. 算術符号

3.1 算術符号の原理

１）　数直線符号化

算術符号は非ブロック符号の代表といえる．今，任意の符号データの先頭に小数点をつけると，0 から 1 の数直線上のある領域を表すことになり，符号系列が n ビットの符号語には数直線上で $1/2^n$ の領域が割り当てられていると解釈できる．これを数直線マッピングという．

さて，情報理論では，確率 p の事象に $\log_2(1/p)$ビットを割り当てるのが理想符号化である．逆に n ビットの符号語を割り当てられた事象については，その確率が $1/2^n$ である場合が理想となる．数直線マッピングでは，n ビットの符号語には数直線上で $1/2^n$ の領域が割り当てられるので，「シンボルに，その出現確率に基づいて数直線上の領域を割り当てる」処理を，数直線符号化と定義すると，数直線符号化においては，「確率 p の事象の数直線上での理想的領域は $(1/2)^{**} \log_2(1/p)$（=p）である」という極めてシンプルな原理が導かれる [10]．

したがって数直線利用符号化ではその対応領域をシンボルの出現確率に応じて分割してゆき，最終的にその領域に存在することがわかるような符号語を出力すればよいことになる．

２）　算術符号化の定義

さて，数直線符号化において途中段階でのシンボル復号を可能とするためには，１つのシンボルの対応領域は連続させてとることが必要条件になる．この結果，数直線利用符号における最小必要符号長 L は該当系列の生起確率（符号化を行う上での想定確率）を P とするとき，

$$-\log P \le L < -\log P + 2 \qquad (1)$$

で与えられる．

通常のブロック符号化では

$$-\log P \le L < -\log P + 1 \qquad (2)$$

となるので，この１ビットの差は１つのシンボルの対応領域は連続させてとるという条件，あるいはそれから必然的に導かれる，「符号割り当てに際し，シンボル系列を出現確率順で並べ替えることが許されない」という条件に起因していると考えられる．ただ算術符号の上限は符号系列全体での冗長度を議論しており，ブロック符号においては通報あたりの冗長度を議論しているため，実際はむしろ算術符号の方がブロック符号より高い符号化効率が期待できると考えてよい．

以上の考えに基づき，数直線利用符号を実用化する

算術符号は以下のように定義できる[10].

［算術符号の定義］
[0,1]の数直線上でシンボルの出現確率に応じてその対応領域を分割してゆき，最終的に選定された区間内に有り，他の区間には含まれない領域を表現するのに必要な座標を符号語とする符号形式を算術符号と規定する．座標については，小数点以下 N 桁の有効数字で表現される 2 進小数γの座標により[γ,γ+2^{-N}]で表現される領域が表現されるものとする．上記定義に従って対応区間に完全に含まれる座標（N bit の符号語γ）を対応区間に対する符号語とすれば復号が可能である．（以上）

3） 正規化とフラッシュ処理

算術符号では，上述の累積確率を精度よく計算する上で，正規化という処理を導入する．これは，符号化が進むにつれ累積確率がどんどん小さくなってゆくので，演算精度を保つために，有効領域の大きさをレジスタサイズで決まる精度の 1/2 以上に回復させる処理である．具体的には，有効領域のレジスタの MSB が 0 になれば，1bit シフトして，領域を等価的に 2 倍にする．これは，符号語段階での 1bit の出力追加に対応する．

正規化のタイミングとしては，領域レジスタの MSB がゼロになった時点で行うビット単位方式が普通であるが，処理の高速化のために有効領域が最大有効値の 1/256 以下になったときに正規化を行うバイト単位方式もある．この場合は，後述の最小確率の設定にも影響が生じる．

さて，上記の定義に従う算術符号において，算術符号の一意復号可能性とは「その後どのような符号データ系列が続いても対応領域に存在することが保証されている」ということである[10]．このとき最終シンボル符号化時のフラッシュビット長（レジスタ内の数直線アドレスデータの出力桁数）は最終正規化有効領域を S とすると(1/2<S≤1)，S の小数点以下高々2bit でよいことが示される．これが前記の数直線符号を終端する際に，生じる損失の具体的証明にもなっている．

3.2 算術符号によるマルコフ情報源符号化

既に述べたように算術符号はマルチコンテクスト情報源の効率的符号化手段として登場した．では何故それが可能であったかというと算術符号では発生シンボルごとに有効領域の大きさとその下限アドレスを計算すればよい．すなわち，情報源拡大を行うブロック符号において必要である「通報の設定とそれに伴う符号語の割り当て」が不要でありそれぞれのシンボルの符号化を他のシンボルの符号化とは独立に行えるというのが算術符号の大きな特徴である．算術符号で情報源拡大が行われているかどうかは若干議論があり，シンボル単位で個別の符号化処理をしていると考えれば情報源拡大ではない。一方最後に符号語を割り当てるという点で系列として情報源を拡大していると考えることもできる．一般には算術符号は情報源の拡大という概念自体がないと考えてよく．またブロック符号での状態分離は不要なためインタリーブ処理やメモリ制御は不要であり，状態混在拡大における複雑な符号設計も不要である．

3.3 算術符号の効果

1） 2 値画像符号化

2 値画像の符号化においては MH,MR[14]などのように変化点に着目した符号化が当初検討されていた．これは文字やグラフなどを符号化対象とした場合極めて有効な手法であり，そのエントロピー符号化ではランレングス符号化やエッジ位置表現の符号化などで算術符号を用いる効果は小さくハフマン符号で十分である．しかし，ハーフトーン画像を対象に加えると一挙に問題が生じる．たとえば 1 画素の白と 1 画素の黒が繰り返すパターンを MH 符号化すると符号量はオリジナルの 4.5 倍になる．MR で 1 画素単位の白黒市松模様を符号化するとオリジナルの 3.75 倍，MMR で 1 画素の白と 1 画素の黒が繰り返す 1 ラインと全白の 1 ラインのペアが繰り返されるとオリジナルの 4 倍の符号量になる．このような情報源に対してはマルコフモデルをベースとし，1 画素あたりの最悪の符号量をほぼ 1 ビットにできる算術符号化は極めて有効である．JBIG-1 においては floating 画素を定め組織的ディザへの有効性を高めているが，誤差拡散などのハーフトーン画像でも原画そのものより符号量が大幅に上回るケースは少ない．ちなみに QM-Coder の最悪符号量についてはオリジナルの 7/6 倍になるシーケンスが効率の悪い例として指摘されている[15]．また低 LPS 推定確率状態において LPS 発生で LPS 推定確率があがってもその後の MPS 発生で直ちに LPS 推定確率が下がる状態に移動することに伴う不具合の可能性については MQ-Coder で解決された[16]．

2） 多値画像符号化

JPEG 符号化では DCT 変換後の係数を量子化し DC 係数については差分を符号化し AC 係数については 0 ランのランレングスと非ゼロ係数とをまとめてハフマン符号化する．この両係数の符号化においては算術符号化も定義されておりハフマンよりも高い性能が得られるが十分な比較検討は行われていない．なお，ロスレス符号化やプログレッシブ符号化，ハイアラーキカル符号化などについてもハフマン符号と算術符号とが定義されている。これらはいずれも算術符号向きといえるがベースラインに含まれていないこともあり十分な検討が行われていない．

次に JPEG-LS では算術符号の選択が大きな議論となった．これはロスレス符号化のためマルコフモデル符号化が適しており，特に CG や文書画像では予測の一致率が極めて高いため予測誤差を符号化する場合に予測誤差がゼロの確率が 1/2 を超えることがあり，コンテクストを認識しつつ情報源拡大を行うことはハフマン符号では困難であったためである．しかし議論の結果，必須機能である Part-1 ではモード分離機能を導入し自然画モードでは予測誤差信号の非拡大ハフマン符号化とし，CG・文字モードでは予測一致・不一致の 2 値情報を情報源拡大符号化の一種である MELCODE で符号化したあと，予測不一致の場合には拡大を打ち切って予測誤差信号をハフマン符号化することとなった．なお，CG・文字モードでは予測が一致し続ける限り CG・文字モードを継続する．Part-2 では算術符号化がただ一つのエントロピー符号化手段であり，モード分離は不要となってコンテクストごとに予測誤差信号を符号化する．なお，Part-2 の算術符号化は JPEG で使用されている QM-coder ではなく変形した算術型 MELCODE[17]を使用している．これは Part-2 では必ず Part-1 を圧縮率で上回ることを目指したことから，より効率の高い算術符号を必要としたためである．

JPEG 2000 においては EBCOT で DWT 係数のビットプレーン符号化を行っており，どちらが先ともいえないが 2 値情報の符号化に帰着されるため 2 値算術符号化が唯一のエントロピー符号化手段である．JPEG 2000 では JBIG-2[18]と共に MQ-coder が採択されている．

３） 動画符号化

動画標準に初めて算術符号化が導入されたのは H.264 であり，エントロピー符号化としてはハフマン符号の CAVLC と算術符号の CABAC が採用されている．続く H.265 では CABAC のみ，すなわち算術符号のみとなっている．

4. 算術符号の実用化

このように算術符号は高い性能を有するがハフマン符号と比較するとその負荷は大きいため，これまでその実用化に向け様々な簡易化手法の検討が行われてきた．

4.1 シフト型算術符号

算術符号の最初の実用的提案としては LR 符号[1]を挙げることができる．LR 符号では LPS の想定確率を(1/2)のべき乗に制限するものでその結果 LPS の領域計算は有効領域のシフト演算のみで行うことが可能となる．しかしながら想定確率が 1/2 と 1/4 の丁度中間での効率が 95%に低下してしまう．次の谷である 1/4 と 1/8 との丁度中間では効率が 97.2%程度でおさまる．

4.2 減算型算術符号

LR 符号に続いて提案された Q-Coder[19]は LPS の領域幅を有効領域幅によらず固定とするものである．したがって領域計算が不要となる．有効領域幅は最大で 2 倍の範囲を動くので想定確率としては 2 倍の範囲で変動が生じるが(平均的)LPS 確率は自由に取れるので，領域計算がより正確ではあるものの確率が限定される LR 符号よりも平均的に高い性能が得られる．Q-Coder は確率が 1/2 近辺で LPS の領域幅が MPS の領域幅を上回ることがあるのでその補正を行う CE 処理[20]を加え QM-coder や MQ-coder として標準に採用されている．

CABAC は減算型算術符号であるが有効領域のサイズを 4 通りに分けてそれぞれの LPS 幅を定めている．すなわち,最大有効領域の 5/8 未満,5/8 以上 6/8 未満，6/8 以上 7/8 未満，7/8 以上である．それぞれの場合の有効領域の期待値は最大有効領域の 9/16,11/16, 13/16, 15/16 であるため，各 LPS 推定確率における LPS サイズの値は有効領域の大きさによらずほぼ 9:11:13:15 となる．ただ LPS 確率の高い方から 3 つのコンテクストでは有効領域の最も小さいケースでの LPS 幅のみを本来より小さくし 128（最大有効領域の 1/4）で固定としている．したがってこれらのケースでは LPS 幅と MPS 幅の逆転は生じない．LPS 確率が高い方から 2 つのコンテクストでは 5/8 以上 6/8 未満，6/8 以上 7/8 未満，7/8 以上の有効領域において MPS/LPS サイズの逆転が生じうることになる．表 1 に 64 状態のうち LPS 確率の高い方から 8 状態のみの LPS 幅を示す．

表 1　CABAC の LPS 幅テーブル（抜粋）

State	Quant CodeIntRange			
	0	1	2	3
0	128	176	208	240
1	128	167	197	227
2	128	158	187	216
3	123	150	178	205
4	116	142	169	195
5	111	135	160	185
6	105	128	152	175
7	100	122	144	166

4.3 算術型 MELCODE

算術型 MELCODE[17]はブロック符号である MELCODE を算術符号として再構成したもので，ブロック型 MELCODE ではそのパラメータが次数(整数値)であり，次数から LPS 領域幅が求まるのに対し算術型 MELCODE のパラメータは LPS 領域幅で与えられる．したがって算術型 MELCODE では非整数次数の

MELCODE が定義できることになる．また有効領域幅が「最大有効領域幅の 1/2＋想定 LPS 領域幅」より大きい場合は減算型符号として振舞うがそれより小さいと LPS 領域幅が規則的に小さくなり，本来の算術符号に近い振る舞いをするので減算型で生じる LPS 確率が 0.5 付近での効率低下がみられない．

4.4 状態遷移型算術符号

ハフマン符号では符号ツリーをたどることで符号化／復号が行える．これは符号ツリーのノード（節）アドレスを遷移先とする状態遷移として符号化／復号を記述することであり，符号が確定する場合は出力符号語を記述し，遷移先はルート（根）に戻る．したがって算術符号を簡易化する上でハフマン符号と同様に状態遷移によって算術符号の動作を記述できるようにすることが簡易化につながるといえる．筆者らが提案している STT-coder[21]はこのような考えに基づいている．

たとえば 6 ビットのレジスタをもつ算術符号で考えると最大長の符号は 6 ビット(確率 1/64)であり，最小長の符号は（複数シンボルに対し）1 ビット(確率 1/2)である．6 ビットの符号は確率 1/64 の場所が数直線上のどの位置にあっても割り当てることができるがそれより短い符号はその対応確率をもつ領域が数直線上に存在する位置が制限され，同じ確率でも存在位置によっては表現する符号長が 1 ビット伸びる．これが算術符号のもつ制約であり，その制約が前記の式(1)に対応する．

通常の算術符号は P を全系列に対して求めることで L と log(1/P)との差を無視できるようにしており，このため有効領域の下位アドレスを記憶しておく必要がある．また桁上がり制御の問題もここから生じる．

算術符号を状態遷移で記述するには下位アドレスを常にゼロにする必要がある．しかし LPS 領域が上位配置の場合，MPS 発生では常に下位アドレスがゼロのままで問題がないものの LPS 発生では LPS と MPS の境界位置に有効領域の下位アドレスが移動するため，その位置に対し制限が生まれ，LPS 幅がとりうる値に制約が生じてしまう．そこで有効領域の下端アドレスと確定済み符号の与えるアドレスとの差をオフセットと呼び，オフセットの値としてゼロ以外も許容すればオフセットの種類数に応じて記憶すべき状態遷移のテーブルサイズが増大するものの許される LPS 幅の自由度が大きくなるため広範囲の情報源に効率よく対応できる．しかしながらオフセット値が大きいと有効領域の期待値は小さくなるので必ずしもオフセットの種類を増やすことで符号化効率の上昇につながるともいえない．その結果，たとえば 6 ビットシステム（有効領域幅 64）では効率が最大であるオフセットの種類数は 4 でその時のオフセット値の組み合わせは(0,16,24,28)となることが示されている．このとき効率の低下は理想算術符号に比べ 0.5%程度であり，充分実用的であると判断できる．

また，この結果の示す規則性はレジスタ長をさらに拡大したときの設計指針としても有効であると考えられている．

5. 算術符号の今後の展開

5.1 Raw 符号との融合

算術符号の簡易化として減算型符号が有効であることは確かである．しかしながら LPS 確率が高い場合（LPS 幅が 1/4 以上）は有効領域が小さい場合に MPS/LPS のサイズの逆転が生じる．QM-coder や MQ-coder の CE はこれを補正するものであるが LPS 確率が丁度 1/2 の情報源では補正ができない．このため LPS 確率が丁度 1/2 の情報源への対応が残る．その一つの対策が Raw 符号（オリジナルデータ）との融合[22]である．LPS 確率が丁度 1/2 では LPS，MPS に各 1 ビットのブロック符号を割り振れば効率は 100%である．したがってその符号を算術符号の系列中に埋め込めば算術復号が不要となるうえ，効率も上がる．埋め込み方としては受信側でそのコンテクストに出会ったときに，次に読み込むべき算術符号の先頭が Raw 符号となるよう配置すればよい．なお，算術符号に対し誤解を与える解説として,「算術符号では復号処理の終了点が判別できない」という表現がされていることがあるがハフマン符号でも算術符号でも必ずデータの読み出しが必要になるタイミングがあるので，読み出そうとしてデータがなければ復号処理は終了する．また復号されたデータに無効な MPS が含まれる可能性は算術符号でも情報源の拡大を行うハフマン符号でも同様に存在する[23,24]．また「算術符号は数直線のアドレスを送るので，下位の 0 は省略できるため，高い圧縮率が得られる」という説明がされていることもある．確かに何らかの手段で，符号データの終了がわかり，かつ本来予定されているシンボル数の復号が終了していない場合に(下位アドレスとして省略されたであろう)0 を符号データと仮想して読み込むことは可能である．実際に JBIG-1 においてもそう規定している.しかし，この仮想データは，事前に取り決めてさえおけば，実は 0 でも 1 でも可能である．また，そう取り決めてさえおけば，たとえばブロック（ハフマン）符号においても同様に適用可能であることはいうまでもない．なお効率よく符号化されたデータは 0 と 1 がほぼランダムになると予想されるので仮にアドレスの最後の 0 を省略してもそれによる符号長短縮の期待値は 1bit にすぎない．

5.2 多値算術符号

実用化されている算術符号の多くは 2 値であり，多値算術符号の検討は少ない．まず多値算術符号と多値のブロック符号とを比較すると最尤シンボルであっても多値のブロック符号では 1bit 以上の符号語が割り振られるので最尤シンボルの出現率が極めて高い場合に算術符号の効果がある．また発生頻度が第 2 位のシンボルについても 3 値以上の情報源のブロック符号では少なくとも 2 ビットの符号語が割り振られるので第 2 位のシンボルの出現確率が 1/2 に近い場合にも算術符号の効果が大きい．

次に多値算術符号を 2 値算術符号と比較すると 2 値情報源への分解が不要で高速化が期待できることがあげられる．一方その問題点は最稀シンボルの確率がレジスタ長で決まる最低確率より低い場合の効率低下である．また，学習のアルゴリズムも 2 値の例がそのままでは使えないので新たに開発が必要となる．具体的な方針としては前者に対しては多値のレベル数を限定して階層的な多値化を考えるのがよいと思われる．また後者についてもやはり階層的な処理が効果的と思われる．

6. むすび

算術符号が多くの画像符号化標準に取り入れられつつある状況の背景について考察し今後についても展望した．算術符号の本質は各シンボル系列に対し独立に符号割り当てが行えることであり，その効能がマルコフモデル情報源の実用的符号化の道を拓いたといえる．またその高効率性と学習への適応性から本来マルコフモデルを想定していなくてもコンテクスト認知を自動的に行うことでマルコフモデルとしての（予期しない）情報量を実現できることになる．一方でその高速化と LPS 確率が 0.5 の付近での減算型符号の効率化として多値化や Raw 符号との融合についての可能性を示唆した．今後も算術符号の検討が進みさらに普及が促進されることを期待したい．

参考文献

1) G.G. Langdon, J.J. Rissanen: “Compression of Black-White Images with Arithmetic Coding”, IEEE Trans. on Communications, Vol. COM-29, No.6, pp.858-867 (1981).
2) ISO/IEC 11544, “Progressive bi-level image compression” (Dec. 1993).
3) ISO/IEC 10918-1,“Digital Compression and coding of continuous-tone still images: Requirements and guidelines”(Dec. 1993).
4) ISO/IEC 14495-1:“Lossless and near-lossless compression of continuous-tone still images - baseline” (Dec. 1999).
5) ISO/IEC 14495-2:“Lossless and near-lossless compression of continuous-tone still images – Extensions” (Mar. 2002).
6) ISO/IEC15444-1:”JPEG2000 Image Coding System: Core coding system”(Dec.2000).
7) ITU-T H.264 | ISO/IEC 14496-10 Advanced Video Coding (2014).
8) ITU-T H.265 | ISO/IEC 23008-2 High Efficiency Video Coding (2015).
9) 小野 : “マルコフ情報源のエントロピ符号化－算術符号化処理の必然性と MELCODE への適用” 画像電子学会誌 Vol.21,No.5, pp.475-485(1992.10).
10) 小野 : “算術符号とその画像符号化への適用に関する考察” 画像電子学会誌 Vol.31,No.5,pp.745-754 (2002. 9).
11) 大西,上野,小野 : “2 値情報源の符号化圧縮”, 信学論 vol.60-A, No.12, pp. 1114–1121(1977.12).
12) S. W. Golomb: “Run-Length Encodings” Trans. IT, IEEE, pp. 399-401, (July 1966).
13) 大西,上野,小野他: “予測変換信号のスタートパターン別ランレングス符号化方式”信学総全大 1016（1977）
14) R. Hunter, A.H. Robinson: “International Digital Facsimile Coding Standard”, Proc. of IEEE, Vol.68, No.7, pp.854-867 (1980).
15) 小野，木村他 :“QM-Coder における最悪符号長の検討” 信学春大 D-273（1994）.
16) 木村，吉田，小野 :“MQ-Coder における最悪符号長の検討” 信学情報・システムソサイエティ大 D-11-44（1998）.
17) 小野，木村他 :“算術符号型 MEL-CODE” 信学春全大 A-152（1989）.
18) ISO/IEC 14492, Lossy/lossless coding of bi-level images” (Dec. 2001).
19) W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, Jr., R.B. Arps: “An Overview of the Basic Principles of the Q-Coder, Adaptive Binary Arithmetic Coder”, IBM Journal of R&D, Vol.32, No.6, pp.717-726 (1989).
20) 小野,吉田,木村,木野 :“減算型算術符号における符号化効率とその向上方式” 画電全大 3(1990).
21) 上野，小野 :“状態遷移テーブル参照型算術符号 STT-coder の設計” 画像電子学会誌 Vol.43, No.1, pp.62-70, (2014.01).
22) F. Ono: “Improvement of Table driven type arithmetic code by combining raw code”, 4A-1, IEVC2010, (2010).
23) 小野，木村他 :“算術符号化における復号終了条件と復号シンボルの有効判定” 信学秋大 D-198（1993）.
24) 木村，小野他 :“情報源拡大符号化における終了記号符号化による有効シンボル通知” 信学秋大 D-199（1993）.