博士論文

状態遷移テーブル参照型算術符号に関する研究

２０１４年３月

東京工芸大学大学院

工学研究科 電子情報工学専攻

上野幾朗

梗概

算術符号は高い圧縮性能と様々な情報源に適用できる汎用性を有する.しかし,その演算負荷が高いため，如何に効率を保ったまま演算負荷の低減を行うかという検討が長年進められてきた．その結果，最近では，算術符号が多くの画像／映像符号化標準に採用されるようになっている．しかしながら依然としてその負荷は大きく，算術符号の普及の妨げとなっていると考えられる．これまでの算術符号の簡易化検討の殆どはシンボルの確率に応じて対応領域を割り当てる領域計算方法を対象としている．これに対し本論文では算術符号では必要であるがハフマン符号では不要である領域下端アドレスの計算に着目し,ハフマン符号と同様に,状態遷移先と符号とを出力する状態遷移テーブルを参照して算術符号化処理が実行できるようにすることを検討した．以下では，このような処理の簡易化を実現した 2 値算術符号器 STT-coder の提案と評価を行う.

最初に，最も簡易なレジスタ長 3 ビットの構成を例にとり，提案方式の動作と符号化性能を検証した．その際，領域下端アドレスの計算を不要とする手段としては最も簡単な，再正規化時（領域が 1/2 以下に減少した時）に必ずフラッシュ（符号確定の処理）を行う手法（方式 1）を考察した．この時，後続のシンボルと組み合わせて有効領域を再分割することにより，フラッシュに伴って生じる効率低下を抑止する拡大フラッシュの概念を導入した．さらに，若干のテーブルサイズの増大を許容し，出力可能な符号が確定した時にはフラッシュではなく通常の再正規化を行うように修正した簡易化手法（方式 2）を提案した．方式 2 では，有効領域のアドレス下端と確定済み符号の与えるアドレスとの差を記憶する必要があるためこの値をオフセットと定義した．このような基礎検討に基づき実用性の高い算術符号を設計するためレジスタ長を 6 ビットに拡張し，テーブルサイズを支配する設計パラメータであるオフセット数に着目し,その最適化を検討した.そのために，シンボル発生確率の偏りが異なる各種 2 値情報源（MPS 確率が 0.5 から 0.987 程度）に対し理想的には約 99%の効率が維持される 8 つの情報源モデル群（確率状態）を想定した．オフセットの許容値により，生じ得る有効領域幅に対する確率状態ごとの両シンボルへの領域割り当て幅の自由度が制限される.そこでオフセット値の許容種類数 N を 1 から順に増加させ, N に応じた最適なオフセット値の組み合わせをグリーディ法により求めることとした．その結果，符号化効率については N が大きいほど低 MPS 確率の確率状態での符号化効率を高

くできるものの，高 MPS 確率の確率状態での効率は，オフセット値が大きいほど有効領域の期待値が小さくなることにより,逆に低下することを明らかにした.この結果，対象範囲の情報源に対する平均的効率の最大化を実現するうえで最適なオフセット数が存在することを明らかにすることができ，6 ビットシステムでは N=4 が最適で,そのオフセット値 D の組み合わせは(D=0,16,24,28)となることが導かれた. さらにオフセットの大きな D=28,24 においては現時点での符号化の最適化よりも将来の有効領域の増大化を優先させることにより，広範囲の確率状態に対しより良好な符号化性能の実現が可能であることが示された．このような条件下における静的符号化効率は，理想算術符号に対し，MPS 確率が低い範囲では 0.5%程度の低下に留まり，充分実用的なレベルにあることを明らかにした.

次に STT-coder の確率推定（確率の学習）問題について考察し，その符号化性能（動的性能）について解析した．確率推定方式としては，簡易化が容易な状態遷移型確率推定を前提とした上で,これまでの課題であった LPS 確率が 0.5 付近での性能低下の原因を究明し，広範囲の情報源に対して符号化性能の制御を可能とする「遷移 LPS 比率」の概念を新たに導入した．確率状態が 8 状態で理想符号化を前提とした，提案学習方式の動的性能を解析したところ，遷移 LPS 比率の導入により最低符号化効率は導入前より約 2%向上した. さらに確率状態の区分は 8 種類のまま，遷移の過程を示す 1 ビット情報を追加した 16 状態化を行うことで動的性能はさらに約 4%向上することが示された．これを既存の標準算術符号である MQ-coder で同程度の情報源範囲に対応させた 22 状態版と比較したところ，STT-coder の符号化効率は平均的に約 1.5%上回り，従来方式に対する動的性能の改善が確かめられた.

以上のように,本論文ではテーブル参照型算術符号 STT-coder を新たに提案し，算術符号でありながら簡易なテーブル参照により符号化処理が実現できること，既存の標準算術符号化に比べて同等以上の符号化性能が得られることを明らかにした.

目次

第1章　序論

1.1　本研究の背景と目的

算術符号は高い圧縮性能と様々な情報源に適用できる汎用性を有する.しかし，その演算負荷が高いため，如何に効率を保ったまま演算負荷の低減を行うかという検討が長年進められてきた．その結果，最近では，算術符号が多くの画像／映像符号化標準に採用されるようになっている．しかしながらハフマン符号[1)]と比較すると依然としてその負荷は大きく，算術符号の普及の妨げとなっていると考えられる．そこで本研究では，ハフマン符号と同様に状態遷移先と符号とを出力する状態遷移テーブルを参照して算術符号化処理が実行できるようにすることにより算術符号化の簡易化を図る.

1.2　従来の研究と本研究の位置づけ

エントロピ符号化として従来から広く使われてきたハフマン符号化では，発生する通報（拡大情報源シンボル）に対してテーブル形式の符号語が割り当てられるため，各通報に対して整数単位の符号長が定まる．それに対し，算術符号化では情報源シンボルの発生確率に応じて算術演算により符号を生成するため，符号長をより精密に小数で表現が可能となる.このため特に 1/2 を超える発生確率（理想符号長 1 未満）を有する情報源に対しても高い符号化効率が実現可能である．また，ハフマン符号化では対応が困難なマルチコンテクスト符号化への対応が容易であることから，マルチコンテクストで情報源拡大が必要な場合に威力を発揮する.

そのため，算術符号は JPEG2000[2),3),4)]，MPEG-4 AVC[5)]，HEVC[6)]など多くの画像／映像符号化標準に採用されている．ただし，情報源入力に対しテーブル参照形式で符号生成が可能なハフマン符号に比較して，算術符号化の演算負荷は高いため，これまでレンジコーダ[7)]，Q-coder[8)]や QM-coder[2),9)]など演算負荷低減を目指した算術符号化の検討が数多く進められてきた．ただし，これまでの算術符号の簡易化検討の殆どはシンボルの確率に応じて対応領域を割り当てる領域計算方法の部分を対象としている．これに対し本論文では算術符号では必要であるがハフマン符号では不要である領域下端アドレスの計算に着目し，ハフマン符号と同様に，状態遷移先と生成符号とを出力する状態遷移テーブルを参照して算術符号化処理が実行できるようにすることを検討している．本論文では，このような

処理の簡易化を実現した2値算術符号器STT-coderの汎用的な提案とその性能の評価結果について述べている．

次に，算術符号化の重要な要素である確率推定（確率の学習）問題について考察し，その符号化性能（動的性能）について解析した．2 値情報源を対象とすると，確率推定の方法としては，予め定めたシンボル発生確率に応じた複数の状態を用意しておき，出現シンボルの発生に基づき状態間の遷移を行う方式（状態遷移型確率推定）と，MPS（優勢シンボル）の出現度数と全シンボルの出現度数をカウントし，その除算結果を利用して MPS 確率を推定するベイズ型確率推定とが考えられるが，本論文では簡易化に適した状態遷移型の確率推定を前提に議論を進める．ここで，従来の状態遷移型確率推定では，LPS の発生では必ず MPS 推定確率を 1 段階下げ，MPS の発生についてはある割合で MPS を選別し，その MPS（遷移 MPS）が発生した場合のみ 1 段階上げる方式がとられているが，これら従来の手法では，発生確率 0.5 付近での性能低下が課題であった．本研究では，簡易化を前提に，LPS においても一部の LPS（遷移 LPS）でのみ状態遷移を行う手法を新たに導入することにより，符号化効率の向上を図る．

1.3　本論文の構成と概要

本論文の第 1 章では本研究の位置づけおよび目的が示されている．第 2 章では過去の算術符号の提案についてその特徴，性能がまとめられている．

第 3 章では状態遷移テーブル参照型算術符号　STT-coder について最も原始的な 3 ビットレジスタの場合を例にとりその原理を紹介している．提案方法の内，より簡易な方式 1（再正規化時点で常に符号確定）はレジスタ長の拡大や装置規模の拡大に対する汎用化が困難と考えられるため，再正規化時点で最上位ビットが確定しているときは通常の再正規化を行い，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して記憶する方式2を採用するに至っている．

第4章ではレジスタ長を6ビットとしたSTT-coderの設計方針について述べている．まず情報源としては6ビットレジスタでカバーできる情報源の範囲に対して MPS 確率の相違する 8 個の情報源モデルを想定すれば，その理想算術符号での効率は 99%に達することを示している．次に第3章で定義したオフセットの種類数が遷移テーブルのサイズを支配するパラメータであることからオフセットの種類数 N を 1（オフセット値 0）から順に増加させ符号化性能の変化を調べてい

る．具体的にはNが2の時のオフセットの追加値は何が最もよいかを調べるために，すべての追加オフセット値について符号化性能を求めている．その結果追加すべきオフセット値の最適化がなされ，次にN=3の場合の最適化はN=2の最適化例に対し追加すべきオフセットの値をすべて調べるという手法をとっている．このような探査の結果，オフセットの種類数を増やしていくとMPS確率の低い情報源の効率は向上するものの，逆にMPS確率の高い情報源では効率が低下するという現象がみられた．この結果，オフセットの種類数には最適値があることがわかり，その種類数は4であり，その場合のオフセット値の組み合わせは(0,16,24,28)であることを導いている．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た．また，この結果が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る．

続く第5章で確率推定を伴う符号化（動的性能）の考察を行っている．確率推定の基本はシンボルカウンタなどの不要な状態遷移型とし，まず従来の考え方による符号化性能を検証した．その結果，低MPS確率情報源での効率低下が観測された．この現象は従来からの課題であり，その解決を求めて考察した結果，従来方式では必ず状態遷移を行っていたLPS（劣勢シンボル）の発生についてもある比率で状態遷移を行わせることが有効であることを導いた．この「遷移LPS比率」導入という着想はSTT-coderに限らず，他の状態遷移型確率推定にも有効な新たな発見である．この考えに基づき6ビットSTT-coderの動的性能を調べたところ，低MPS確率情報源における効率低下が解消され，性能は情報源シンボルの出現確率によらずほぼフラットにできた．しかし8状態のままでは動的な性能が必ずしも満足できないということから遷移状態数の増加が必要と考えられた．静的符号化との共通性という観点から確率状態の区分は8状態のままが望ましいため，そこに遷移の過程を示す1ビットを付加した16状態遷移での符号化性能を調べたところ，確率区分が16状態での符号化における性能と比べ遜色がないことが示された．また同程度の状態数を有する既存の動的符号化方式と比較した結果，性能の改善が確認され極めて効率的な動的符号化方式の設計が導けたと判断できた．この動的符号化性能の実現においては確率変数の導入が必要である．そこで第6章で検証を行い，確率変数として有効領域幅の統計分布を手掛かりとする場合の具体的評価を行っている．

以上のように本論文は算術符号化の簡易化に新たな観点から取り組み，その最適化を考察し，十分な実用性があることを示したこと，動的符号化について汎用

的な考察を行い従来の問題点の解決策を見出したこと，それらを組み合わせて新たな実用的算術符号化の提案を行ったという点に意義があるといえる．

第2章　算術符号の概要

2.1　算術符号の原理

算術符号の原理は 1950 年代に Elias によって提案[10]され，それを Langdon, Rissanen らが実現可能な形に再編することにより実用化に至っている．算術符号では，符号化シンボルの生起確率に応じて確率区間[0,1)を再帰的に分割し，得られた区間に含まれる点の座標を他の区間と区別できる精度の２進小数で表したものを符号とする．図 2-1 に符号化シンボル０１００を対象とした２値シンボル符号化の概念図を示す．第１シンボルの符号化時には，確率区間[0,1)を０，１の発生確率の比で分割して，A(0)と A(1)の確率区間を作る．ここで，第１シンボルが０であれば領域 A(0)を，１であれば領域 A(1)を有効領域として選択する．次に第２シンボルでは,第１シンボルで選択された確率区間を第２シンボルにおける０，１の発生確率に応じてさらに A(00)，A(01)のように分割する．以下同様に，発生したシンボルに対応する領域を再帰的に分割し，最終的には，最後のシンボルに対応する領域に含まれる座標を他の区間と区別して表現するのに必要な精度の２進小数を符号とする処理を行う．

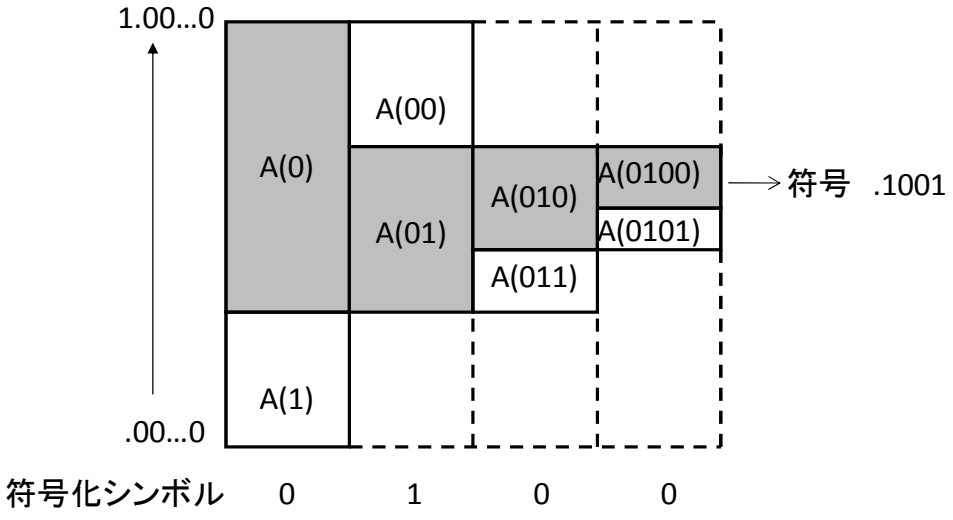


図 2-1　算術符号の概念図

ここで上記手順を数式で示すと以下のようになる．現在の有効領域幅を A，有効領域の下界値を C，シンボル 0 の発生確率を P_0 ，シンボル 1 の発生確率を P_1 とすると

シンボル 0 の領域幅　$A_0=A\times P_0$

シンボル 1 の領域幅　$A_1=A\times P_1$

で表される．シンボル 0 を有効領域の下方，シンボル 1 を有効領域の上方に配置すると，有効領域幅 A，有効領域下界値 C は，以下のように更新される．

・シンボル 0 発生の場合

$A\rightarrow A_0$　　$C\rightarrow C$

・シンボル 1 発生の場合

$A\rightarrow A_1$　　$C\rightarrow C+A_0$

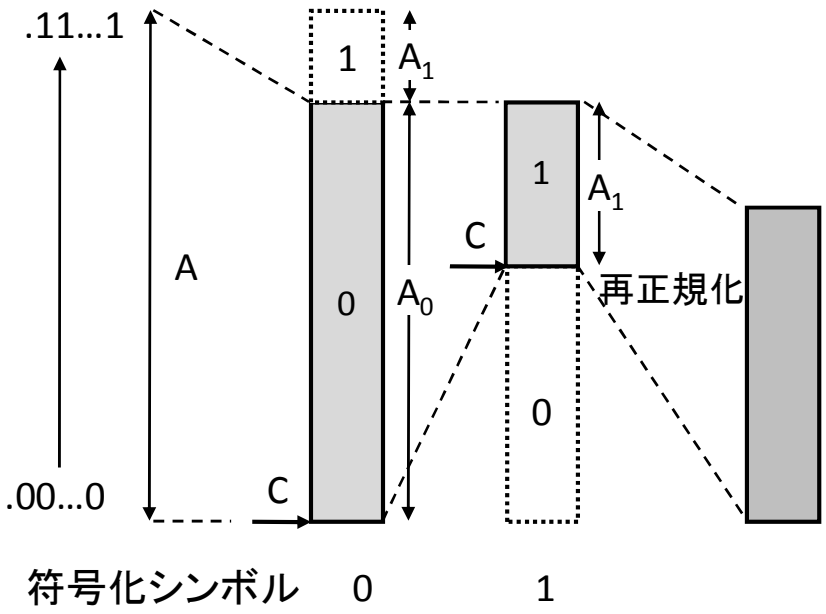


図 2-2　算術符号の領域分割

2.2　従来の算術符号化

上記アルゴリズムは原理的にはシンボル系列が長くなれば，符号長はエントロピに近づく．しかし，領域幅の演算に有効領域幅とシンボル発生確率の乗算を伴うため演算桁数がシンボル系列長とともに増大してゆくことなどの理由により当初実用化には至っていなかった．その問題点を解決するために Langdon，

Rissanen らは LPS の発生確率を 2 のべき乗で近似する手法(LR 型算術符号)[11]を提案し，実用化への道が開けた．また，領域分割の過程で有効領域幅が所定の値（通常は全領域幅の 1/2）より小さくなると，有効領域を拡大（左ビットシフト）するとともに，座標値の上位ビットで確定部分は符号として出力する再正規化を導入し，決められたレジスター長で有効領域幅の精度確保を可能とした．

その後，LPS の領域幅を有効領域の大きさに依らず一定とし，テーブル参照により求める（減算型算術符号）Q-coder[8]，さらに減算型算術符号で LPS 領域が MPS 領域より大きくなった場合の効率を改善するよう，両者の領域を入れ替える条件付 LPS/MPS 交換処理を導入した QM-coder[12),13),14)]及び MQ-coder[2),9)]，MPS 領域が 1/2 以下の場合には，残る領域と 1/2 の差の半分を固定値(LSZ)から減じた値を LPS 領域とする算術型 MELCODE[15),16)]など領域幅の算出に必要な演算を簡易する方式が提案されている．これらの特徴を表 2-1 にまとめた．なお本論文では，簡易化に適するよう，符号化シンボル０，１を MPS（優勢シンボル），LPS（劣勢シンボル）に変換した上で扱っているが，表 2-1 に示すようにこれは従来方式でも共通である．ただし，従来方式においては有効領域幅 A の演算簡易化を主眼としたものであるのに対し本研究では，下界値アドレスの演算を行わず，有効領域幅および領域分割後の状態をテーブル参照により得ることにより算術符号化を実行するという簡易化手段の実現を検討する．

表 2-1　各種算術符号の領域分割演算比較

技術	適用 算術符号	処理方法	領域演算 A：有効領域幅 A_L：LPS 領域幅 A_M：MPS 領域幅
LR 型	LR 型算術符号	LPS 確率を(1/2)のべき乗に限定し有効領域幅のシフト演算により LPS 領域幅を算出. ［本来任意である LPS 確率を最も近い(1/2)のべき乗値で近似］	$A_L=A\times 2^{-m}$ (m ビットシフト) $A_M=A-A_L$ (m は LPS 出現確率から定まる整数値)
減算型	Q-coder	有効領域幅によらず, LPS 領域幅を常に LPS 出現確率に対応した固定値(LSZ)とする. ［本来 1/2 から 1 まで変化する有効領域 A を常に 3/4 として LSZ を計算］	A_L =LSZ (n) $A_M=A-A_L$ (n は LPS 出現確率から定まる LSZ のインデックス)

条件付 LPS/MPS 交換処理付減算型	QM-coder
	MQ-coder
		減算型で MPS 領域より LPS 領域の方が大きい時は各々の割当領域を交換する処理を追加
		(図 2-3 参照).
			if (LSZ(n) < A/2)
			A_L = LSZ (n)
			else
			A_L = A− LSZ(n)
			
			$A_M=A-A_L$

MELCODE 型	算術型 MELCODE	LPS に固定値(LSZ)を割り当てた時、残る領域が 1/2 以上あれば減算型で済ませ、1/2 以下であれば、残る領域と 1/2 の差の半分を固定値(LSZ)から減じた値を LPS 領域とする
		(図 2-4 参照)。
			if (A−LSZ (n) > 1/2)
			A_L =LSZ (n)
			else
			A_L =(A+LSZ(n)−1/2)/2
			
			$A_M=A-A_L$

MELCODE 型	JPEG-LS 用算術符号[17),18),19)]	MELCODE型領域分割を用い，8 ビットごとに正規化（レジスタを 255 倍する）	if (A−LSZ (n) > $1/2^w$)
			A_L =LSZ (n)
			else
			A_L =(A+LSZ(n)−$1/2^w$)/2
			
			$A_M=A-A_L$
			(w は有効領域幅から定まる整数値)

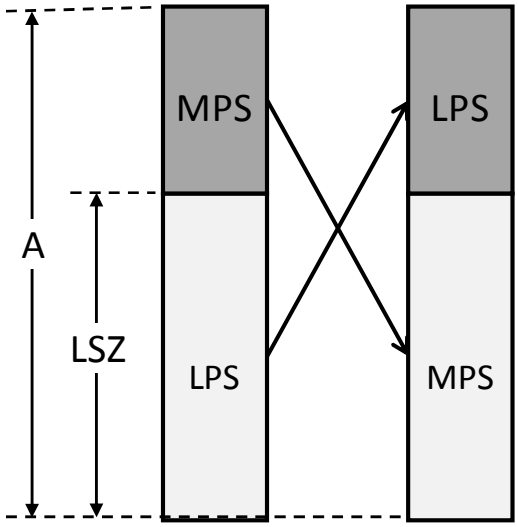


図 2-3　条件付交換処理付 減算型の処理(LSZ(n) ≧ A/2 の場合)

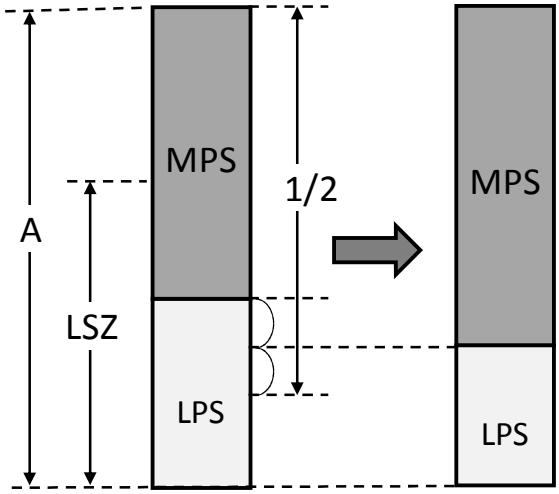


図 2-4　MELCODE 型の処理(A－LSZ (n)≦1/2 の場合)

第3章　状態遷移テーブル参照型算術符号

3.1　まえがき

状態遷移テーブルを参照して算術符号を実行する 2 値算術符号 STT-coder の基礎検討として，本章ではまず符号化レジスタを実用上最も短い 3 ビットに設定して，提案方式の原理を紹介し，達成される符号化性能を検証する[20),21),22)]．最初に，最も簡易な方式（方式 1）として有効領域が 1/2 以下になった時点で算術符号化の再正規化と同時に，常にフラッシュ（符号確定）を行うことを考え，フラッシュに伴う符号化効率の低下を最小限とするための拡大フラッシュ方式を提案する．次に，上位ビットの確定時には通常の再正規化を行い，再正規化時に必ずしもフラッシュは行わないよう条件を緩和することにより，状態遷移テーブルサイズの増大を許容して符号化効率の向上を図る（方式 2）．さらに高 MPS 確率での符号化効率向上を目指し，符号化レジスタ長を 4 ビットにした符号器を 3 ビットの例と組み合わせる形で設計し，その性能を評価する．

3.2　状態遷移テーブル参照型符号

3.2.1　遷移テーブル参照型符号

算術符号（非ブロック符号）と通常のハフマン符号（ブロック符号）を比較すると，その大きな相違はハフマン符号では符号ツリーをたどることで符号化／復号が行えるのに対し，算術符号では一般にそれが困難であることがあげられる．ハフマン符号において符号ツリーをたどるということ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することを本論文ではまず考えることにする．そのためには状態数を実用的な範囲でおさめることが必要であり，実現に向けて簡易化の導入を検討することとする．

算術符号の状態遷移テーブル化を考える上で，通常の算術符号において有効領域下端のアドレスを保持する C レジスタに着目する．C レジスタの必要桁数は桁上がりへの配慮もあって固定化できないため状態遷移テーブル化のネックといえる．そこで有効領域の下端を常に確定・送出済み符号で決まるアドレスそのものにできれば C レジスタの情報保持を不要とでき，ハフマン符号と同様に考えるこ

とができる.これにより,算術符号に特有の桁上がり問題を考慮する必要もない.

以下では検討対象とする符号を状態遷移テーブル参照型算術符号（State Transition Table-Driven Arithmetic Coder：STT-Coder）と称する．最初に最も簡易なレジスタ長 3 ビットでの STT-coder の設計を例にとり，符号化動作を解説する.

3.2.2　再正規化時に常にフラッシュを行う方式（方式 1）

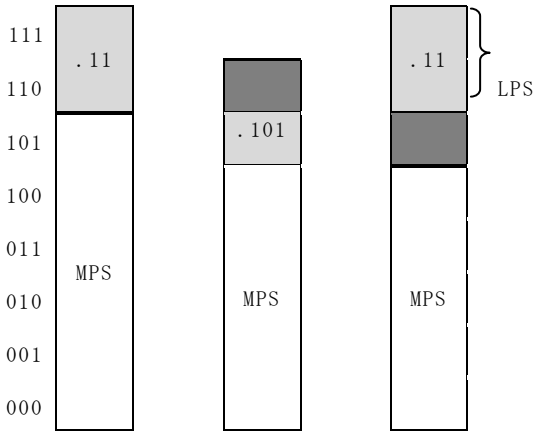
（1）　拡大フラッシュの導入

フラッシュは，符号化シンボル系列の最後で符号を確定する終端処理である．ハフマン符号を算術符号として解釈すると，再正規化の時点で常にフラッシュを行っていると考えることができる．フラッシュに要するビット数は，最終的な有効領域の幅，および下位アドレスに依存するが，算術符号の符号長は，理想符号長+2 ビット未満でおさまる[23]．つまり，与えられたシンボル系列の生起確率を P とすれば，符号長 L は，

$$-\log_2 P \le L < -\log_2 P + 2 \qquad (3\text{-}1)$$

で与えられる．算術符号では通常，系列の最後でのみフラッシュを行うのでこの 2 ビットは系列の情報量に比べて無視しうるものである．しかし，再正規化の時点で常にフラッシュを行うとこの 2 ビットは大きな符号化効率の低下をもたらす.

図 3-1 に,レジスタ長 3 ビットの算術符号で有効領域が 1/2 以下になった場合，つまり再正規化時には必ずフラッシュする例を示す.図 3-1 (a)のように確定した有効領域の確率値が 1/2 のべき乗(=2/8)で，その下位アドレス(.110)が確率値の整



(a)P_L=2/8　(b)P_L=2/7　(c)P_L=3/8

図 3-1　フラッシュ時の領域分割

数倍であれば，フラッシュに要するビット，つまり有効領域を他の区間と区別するための最小ビット長は.110 の上位 2 ビット（確率 1/4 に対応）である 11 となり符号効率のロスは発生しない．図 3-1 (b)のように，有効領域の下位アドレスが有効領域幅の整数倍にならないときにフラッシュを行うと.101 あるいは.110 のどちらを使ってもよいが 3 ビットの符号語が必要となる．この場合，最終的な符号により示される確率領域は，確定した有効領域よりも狭い範囲となる．仮に 101 を符号とする場合，図 3-1(b)の濃いグレー部分が符号語として使われない部分（フラッシュによる符号化効率のロス）となる．フラッシュを行わない場合，当該シンボルの符号化に要する平均符号量 L_0は，LPS の出現確率を P_L，MPS の出現確率を P_Mとすれば，

$$L_0 = -P_L \cdot \log_2(2/7) - P_M \cdot \log_2(5/7) \qquad (3\text{-}2)$$

となるが，フラッシュを行う場合の平均符号量 L_1は，

$$L_1 = -P_L \cdot \log_2(1/7) - P_M \cdot \log_2(5/7) \qquad (3\text{-}3)$$

として表される．

また図 3-1(c)のように有効領域幅が 1/2 のべき乗にならないときにも同様に，最終的な符号.11 により表される領域は，有効領域よりも狭い範囲となり符号化のロスを生じる．

もし，再正規化の時点で領域の大きさが 1/2 のべき乗で，かつアドレスが領域の大きさの整数倍であればフラッシュに伴う符号化のロスは生じないので，逆にこの条件に合うように領域を分解し，後続のシンボルを組み合わせた情報源拡大に相当する領域割当を行い，再正規化すること（以降，拡大フラッシュ）が考えられる．この考えを活用し，領域分割テーブルサイズを最小化したのが，領域分割の方式 1 である．

図 3-2(a)に方式 1 の領域割当のパターンを示す．全領域の大きさが 4 以下ではフラッシュが行われるので領域の範囲は 5～8 で考えている．LPS の割り当ては有効領域の 1/2 を超えないので LPS の割当サイズの可能性は，たとえば領域が 8 なら 1～4，領域が 5 なら 1～2 となる．

図 3-2 では，各領域に対応するシンボルが LPS ならば L，MPS ならば M と記載し，そのシンボルの符号化により符号が確定する場合には，発生する符号をシンボルの後に併記した．濃いグレーで示した領域には，拡大フラッシュが適用さ

れ，細分化したそれぞれの領域に対して複数のシンボル系列が割り当てられる．例えば，領域幅 7，LPS 幅が 3 で LPS が発生した場合（以降，状態 7-3L と表記する），後続するシンボルの結果と組み合わせて領域を 2 分割し，符号を割り当てる．所定の符号を出力した後は，領域幅は初期値の 8 に戻る．

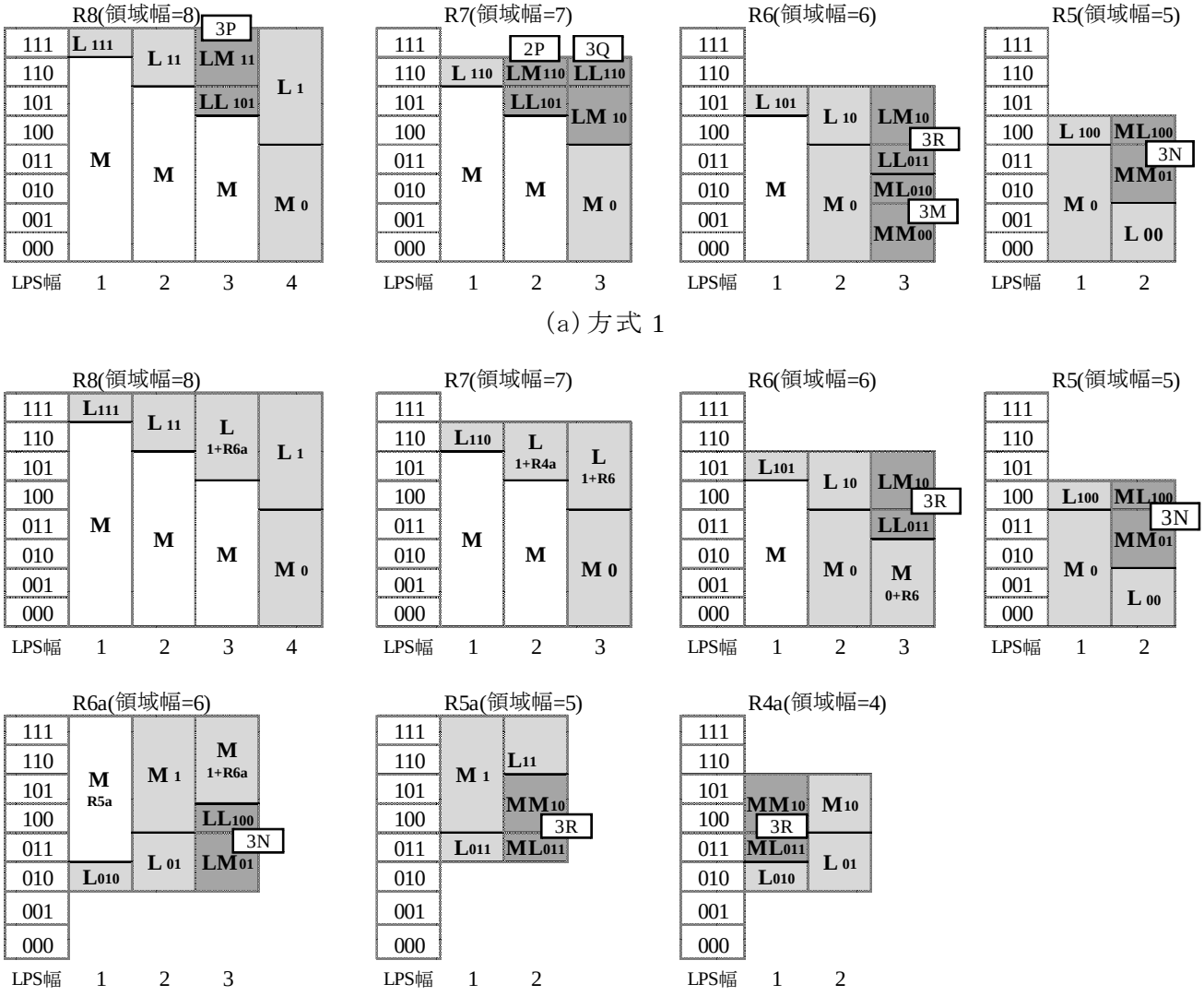


図 3-2　3 ビット STT-coder の領域分割パターン

（2）　状態遷移テーブルによる符号化処理の実現

状態遷移テーブル参照による符号器の構成イメージを図 3-3 に，方式 1 の状態遷移テーブルの内容を表 3-1 に示す．符号器に 1 シンボル入力されるたびに，状態遷移を繰り返すことで，領域分割処理が実現される．符号器への入力として，シンボル（1 ビット，MPS/LPS），LPS 幅（2 ビット，1～4），領域幅インデックス（3 ビット，0～7）が与えられる．領域幅インデックスは，当該シンボル符号化時の全有効領域幅（LPS 領域と MPS 領域の合計）を示すインデックスである．方式 1 では，通常の符号化処理において発生する領域幅 8～5（R8～R5 と表記）と，拡大フラッシュでの 2 シンボル目に対応する領域幅 3 あるいは 2 の場合の 6 通り（3P,3Q,3R,3M,3N,2P と表記）の計 10 種類がある．例えば，状態 8-3L が発生した場合，次のシンボルは図 3-2(a)で 3P と標記された領域幅 3 の状態で符号化する．また，表 3-1 では，3P と 3N，3M と 3R は，同一の領域幅インデックスとして扱い，LPS 幅が固定なので直前に発生したシンボルの MPS/LPS を LPS 幅に代わる入力として両状態を切り換えることにより，実質的な領域幅インデックスの種類を 8 種類に抑えている．

符号器の出力としては，符号／領域幅インデックス（3 ビット） と符号長（2 ビット）とがある．符号長が 0 以外の場合は，符号／領域幅インデックスが，発生する符号語を示し，符号長が 0 の場合は，符号は発生せず，符号／領域幅インデックスが次のシンボル符号化時の領域幅インデックスを示す．表 3-1 では，出力が領域幅インデックスとなる場合には，出力ビットの後に(R7)，(3P)など領域幅インデックスの呼称を付記した．方式 1 のこの状態遷移テーブルの大きさは，320 ビット(=2^6*5)となる．

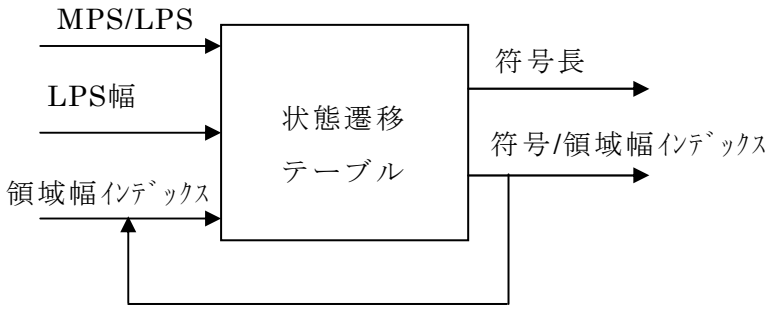


図 3-3　状態遷移テーブル参照による符号器

表 3-1　領域分割テーブル（方式 1）

入力			出力	
3bit	2bit	1bit	3bit	2bit
領域幅 ｲﾝﾃﾞｯｸｽ	LPS 幅	MPS /LPS	符号/ 領域幅ｲﾝﾃﾞｯｸｽ	符号長
111(R8)	1	M	110(R7)	0
		L	111	3
	2	M	101(R6)	0
		L	11	2
	3	M	100(R5)	0
		L	000(3P)	0
	4	M	0	1
		L	1	1
110(R7)	1	M	101(R6)	0
		L	110	3
	2	M	100(R5)	0
		L	011(2P)	0
	3	M	0	1
		L	001(3Q)	0
101(R6)	1	M	100(R5)	0
		L	101	3
	2	M	0	1
		L	10	2
	3	M	010(3M)	0
		L	010(3R)	0
100(R5)	1	M	0	1
		L	100	3
	2	M	000(3N)	0
		L	00	2
000(3P)	1	M	11	2
		L	101	3
000(3N)	2	M	01	2
		L	100	3
001(3Q)	-	M	10	2
		L	110	3
010(3M)	1	M	00	2
		L	010	3
010(3R)	2	M	10	2
		L	011	3
011(2P)	-	M	110	3
		L	101	3

表 3-2　領域分割テーブル（方式 2）

入力			出力		
6bit			6bit		
3bit	2bit	1bit	3bit	2bit	1bit
領域幅ｲﾝﾃﾞｯｸｽ	LPS 幅	MPS /LPS	code/領域幅 ｲﾝﾃﾞｯｸｽ	符号長	outbit
111(R8)	1	M	110(R7)	0	-
		L	111	3	-
	2	M	101(R6)	0	-
		L	11	2	-
	3	M	100(R5)	0	-
		L	000(R6a)	1	1
	4	M	111(R8)	1	0
		L	111(R8)	1	1
110(R7)	1	M	101(R6)	0	-
		L	110	3	-
	2	M	100(R5)	0	-
		L	001(R4a)	1	1
	3	M	111(R8)	1	0
		L	101(R6)	1	1
101(R6)	1	M	100(R5)	0	-
		L	101	3	-
	2	M	111(R8)	1	0
		L	10	2	-
	3	M	101(R6)	1	0
		L	010(3R)	0	-
100(R5)	1	M	111(R8)	1	0
		L	100	3	-
	2	M	011(3N)	0	-
		L	00	2	-
011(3N)	-	M	01	2	0
		L	100	3	-
010(3R)	-	M	10	2	-
		L	011	3	-
001(R4a)	1	M	010(3R)	0	-
		L	010	3	-
	2	M	10	2	-
		L	01	2	-
001(R5a)	3	M	111(R8)	1	1
		L	011	3	-
	4	M	010(3R)	0	-
		L	11	2	-
000(R6a)	1	M	001(R5a)	0	-
		L	010	3	-
	2	M	111(R8)	1	1
		L	01	2	-
	3	M	000(R6a)	1	1
		L	011(3N)	0	-

3.2.3　再正規化条件の緩和（方式 2）

拡大フラッシュは，通常の（非拡大）フラッシュで生じる無駄な確率領域の発生を回避するが，フラッシュを行わない場合と比較すると組み合わされる後続シンボルにおける領域分割の自由度は小さくなる．例えば，状態 6-3M では，後続シンボルについては 2:1 の固定比率で領域が分割されることになる．そのため，その LPS 確率が 1/3 から離れるにしたがい符号化効率の低下が大きくなる．この場合，通常の算術符号のように再正規化ができれば，全領域を大きくできるので分割の自由度が高まり符号化効率の向上が期待できる．そこで，再正規化の場合に必ずフラッシュを行うという方式 1 での方針を変更し，有効領域座標の最上位のビットが確定した場合には，その定まった最上位ビットを符号として出力した上で再正規化を行うことを考える．これを方式 2 と呼び，その領域分割を図 3-2(b)に，状態遷移テーブルを表 3-2 に示す．表 3-2 のテーブルでも表 3-1 と同様に，符号長（2 ビット）で示されるビット数の符号が出力されるが，符号長が 1 の場合には outbit で示される 1 ビットが符号として出力され，Code/領域幅インデックスは，遷移先の領域幅インデックスを表す．なお,「最上位ビット確定時に確定ビットを符号として出力した上で再正規化」という新たな方針を「有効領域が 1/2 以下で拡大フラッシュ」に置き換えたが，無駄な領域の発生抑止に必要である限り拡大フラッシュは維持されていると考えることができる．この再正規化条件の緩和により，状態 8-3L，7-2L，7-3L，6-3M では，領域幅を 2 倍にし，その有効領域座標の最上位ビットを符号として出力することになる．また，再正規化により下位アドレスがゼロ(000)でなくなる場合が生じるのでそのケースについても許容すると新たな領域幅インデックス（R6a，R5a，R4a）を追加する必要がある．図 3-2(b)には，新たに再正規化を行う領域については，対応するシンボルの後に，出力される符号と遷移先の領域幅インデックスを記載した．状態 6-3L，5-3M では，有効領域の最上位ビットが確定していないので，方式 1 同様，拡大フラッシュが適用される．

3.2.4　3 ビットシステムにおける符号化効率

マルチコンテクストの情報源，つまり出現コンテクストにより生起確率が変化する情報源への適用を想定して，3 ビット STT-coder の符号化効率を評価する．情報源モデル（確率状態）として A～I の 9 種類を規定し，各確率状態における有効領域幅 8～4 に対する LPS 領域幅を**表 3-3** のように定めた（拡大フラッシュ時の有効領域サイズ 3，2 に対する LPS 領域幅は常に 1 で固定である）．符号化

時には，与えられたコンテクストのシンボル生起確率に最も近い確率状態が選択され，その時点の有効領域幅に対し，**表 3-3** にしたがって LPS の領域幅が選択される．ここで，提案方式の各コンテクストにおける符号化効率を次のように算出した．

各有効領域で，1 シンボルに要する符号長 l_i (i=8,7,6,5,4,3,2)は，式(3-4)で表される．ここで，P_L(P_M)は当該コンテクストにおける LPS(MPS)の生起確率，A_L (A_M)は選択される確率状態における LPS(MPS)の領域幅であり，A_{ML}は有効領域幅である．

$$l_i = -P_L \cdot \log_2(A_L/A_{ML}) - P_M \cdot \log_2(A_M/A_{ML}) \qquad (3\text{-}4)$$

また，マルチコンテクスト情報源の情報源シンボルの LPS 確率は 0～0.5 の範囲で均一に分布すると仮定して，方式 1，方式 2 それぞれで各有効領域の生起確率 q_i(i=8,7,6,5,4,3,2)をシミュレーションにより算出した．**表 3-3** にその確率を示す．

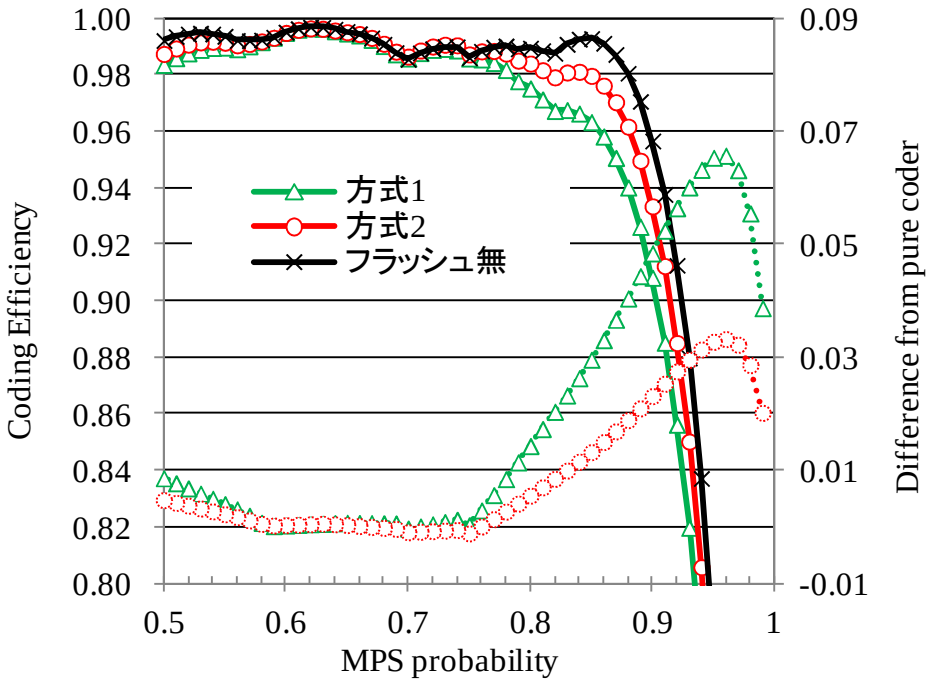


図 3-4　符号化効率（3 ビット STT-coder）

次に，これらから，各コンテクストにおける 1 シンボルあたりの平均符号長 L, 符号化効率 η を次式に基づき算出した．

$$L = \Sigma q_i \cdot l_i \tag{3-5}$$

$$\eta = \{-P_L \cdot \log_2(P_L) - P_M \cdot \log_2(P_M)\}/L \tag{3-6}$$

図 3-4 に，レジスタ長 3 ビットの方式 1，方式 2 の符号化効率，および再正規化をフラッシュを伴わずに行う通常の算術符号化の符号化効率を実線で示す．いずれも最も効率の高い確率状態が選ばれるという前提で効率を表示している．通常の算術符号である再正規化でのフラッシュ無し算術符号化の特性と両提案方式との効率差も併せて点線で示した．

方式 2 は上位ビットが確定した場合には再正規化されるのでフラッシュ無し算術符号と同等の性能が期待されるが，上位ビットが確定しない場合，すなわち有効領域が当初領域の中央値をはさんでいる限りは有効領域がたとえ小さくなっても再正規化できない．このため MPS/LPS シンボルに対する分割の比率が制限され，特に MPS 確率が高い部分では，フラッシュ無し算術符号に比べて符号化効率が劣ることになる．とはいえ，MPS 確率が低い部分では，フラッシュ無し算術

表 3-3　各確率状態の LPS 幅（3 ビット STT-coder）

		有効領域幅						
		8	7	6	5	4	3	2
確率状態	A	1	1	1	1	1	1	1
	B	2	1	1	1	1	1	1
	C	2	2	1	1	1	1	1
	D	2	2	2	1	1	1	1
	E	2	2	2	2	1	1	1
	F	3	2	2	2	1	1	1
	G	3	3	2	2	2	1	1
	H	3	3	3	2	2	1	1
	I	4	3	3	2	2	1	1
q_i	方式 1	.43	.15	.14	.16	-	.12	.01
	方式 2	.40	.14	.20	.18	.01	.06	-

符号とほぼ同等であり，約 99%の符号化効率を達成している．高 MPS 確率以外ではフラッシュに伴う効率ロスは効率的に回避されており，3 ビットのレジスタ長であっても対象とする情報源によっては十分な性能を有するといえる．

方式１と方式２を比較すると，方式１とフラッシュ無しとの差が最大で 7%近くあるが，方式２では最大差が 3%近くに減少していることがわかる．方式２で再正規化の条件緩和により遷移状態数を増加させた効果が現れているといえる．

3.2.5 符号化処理量の比較

STT-coder の符号化処理量を評価するために，国際標準算術符号化 MQ-coder[2),9)]と符号化処理のフローを比較した．図 3-5 に両者のフロー概要を示す．算術符号化の処理は，領域分割，確率推定（確率状態を求める処理．0 で紹介）および生成した符号の出力に大きく分類できる．確率推定は，両者とも簡易化に有利な状態遷移型確率推定を使用しているため処理量として大きな差はない．処理量の差が認められるのは，領域分割，符号出力の部分となる．領域分割において，減算型算術符号である MQ-cder では原則，有効領域幅によらず LPS 幅が決まるが，LPS 幅が MPS 幅よりも大きくなる場合には，両者の幅を入れ替える処理を行う．そのため，有効領域が 1/2 より小さくなるか否か，LPS/MPS 交換処理を行うか否か，など MQ-coder では，加減算だけでなく条件分岐が多数必要となる．それに対し，STT-coder では，常に確率状態，領域幅＋オフセット，符号化シンボルを入力としたテーブル参照で符号，符号化後の領域幅＋オフセットが得られるという簡易な処理である．

また，符号出力に関しては，MQ-coder では桁上がり対策のために生成した符号が 0xFF となった場合には，桁上がりを吸収するために１ビットの０を挿入するビットスタッフィングが必要となる．STT-coder では，確定した符号のみ出力するので，ビットスタッフィングの処理は不要である．このように，STT-coder は従来の減算型算術符号である MQ-coder に比べて，より少ない処理量で符号化が実現できる．

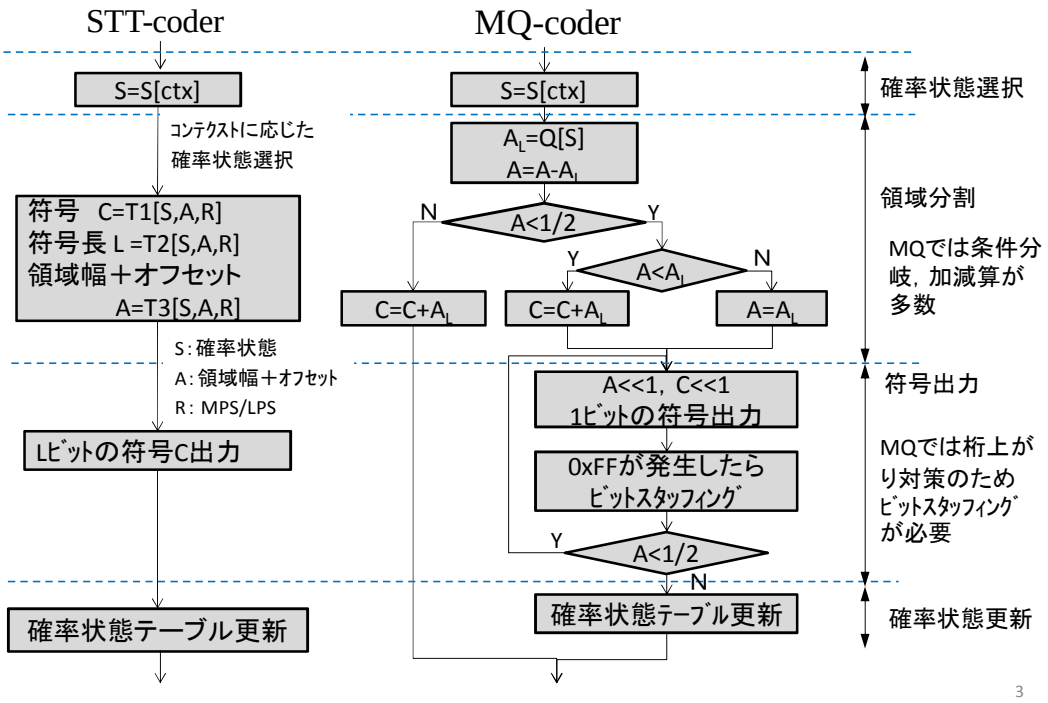


図 3-5　符号化処理の比較

3.3　4ビットシステムの設計

3.3.1　4ビットレジスタ算術符号の設計

前章では，最も簡単なレジスタ長3ビットの符号器を取り上げ，通常の算術符号に対し遜色のない性能が得られることを確認した．しかし，実用的な符号器の実現のためには，ビット長を増大させLPS確率のより低い情報源に対する符号化効率を高める必要がある．一方で，回路規模を抑える(状態遷移テーブルの大きさを抑制する)ため，なるべく状態数を少なくすることが望ましい．そこで，次章でより実用的なレベルである6ビットシステムに拡張する前に，レジスタ長4ビットの場合の符号器を設計し，その振る舞いを調べた．

まず，4ビットレジスタで発生する全領域幅9～16で，最低LPS確率を表現する状態(LPS領域幅＝1)は必ず設定することとした．これにより，低LPS確率での符号化効率向上が見込める．それ以上のLPS確率では，3ビットレジスタの符号器で充分な符号化効率が達成されていることから，3ビットレジスタシステムと同程度の精度で状態数を設定することとした．まず，全領域幅が偶数の場合にはLPS幅が2以上についてはLPS領域幅を偶数のみに限定し，3ビット算術符号の方式2をそのまま踏襲するようにした．次に全領域幅が奇数の場合には，LPS領域幅を奇数のみに限定し，当該シンボルの符号化後に再正規化により他の状態に遷移できる場合は再正規化を行い，できないならば拡大フラッシュを行うこととした．また，11_5(全領域幅＝11，LPS幅＝5)および15_5については拡大フラッシュに伴う符号長の期待値が大きく，これらの状態を省略しても全体性能が変化しないか殆ど向上しないため，設定から除外した．以上の指針に基づいて設計した符号器の領域分割を図 3-6に示す．

3.3.2　4ビットシステムにおける符号化効率

4ビットレジスタ算術符号器の符号化効率を図 3-7に示す．3ビットシステムと同様に，想定する各コンテクストに対する符号化効率を算出し，情報源に対し最も効率の高いコンテクストが選ばれた場合の効率のみを表示してある．3.3.1で説明した4ビット符号器と共に，3ビット符号器の方式2と4ビット符号器フラッシュ無しの場合の符号化特性も併記した．

3ビット符号器と比べると，レジスタの増大により，低LPS確率での符号化効率が向上していることがわかる．またLPS確率が高い部分では，3ビットと同程度の状態数を保っているため，ほぼ同じ程度の符号化効率を達成している．4ビット符号器でフラッシュ無しの符号との効率の比較では，低LPS確率で符号化効

率が劣るが,これは 3 ビットシステムの場合と同様に拡大フラッシュにおいて 1:2 の固定比率で領域を細分化して後続シンボルを符号化する影響によると考えられる.

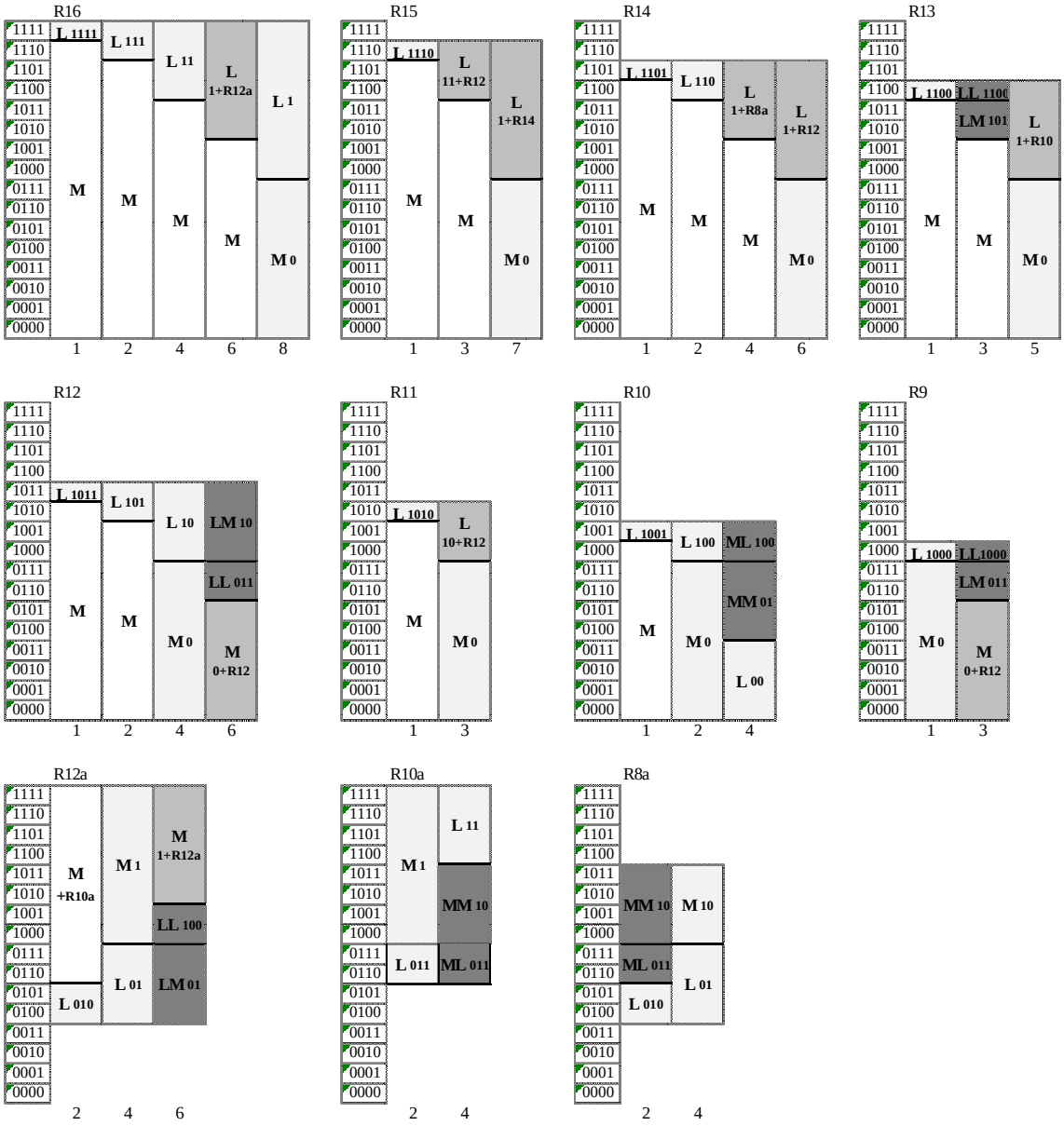


図 3-6　4 ビットシステムの領域分割

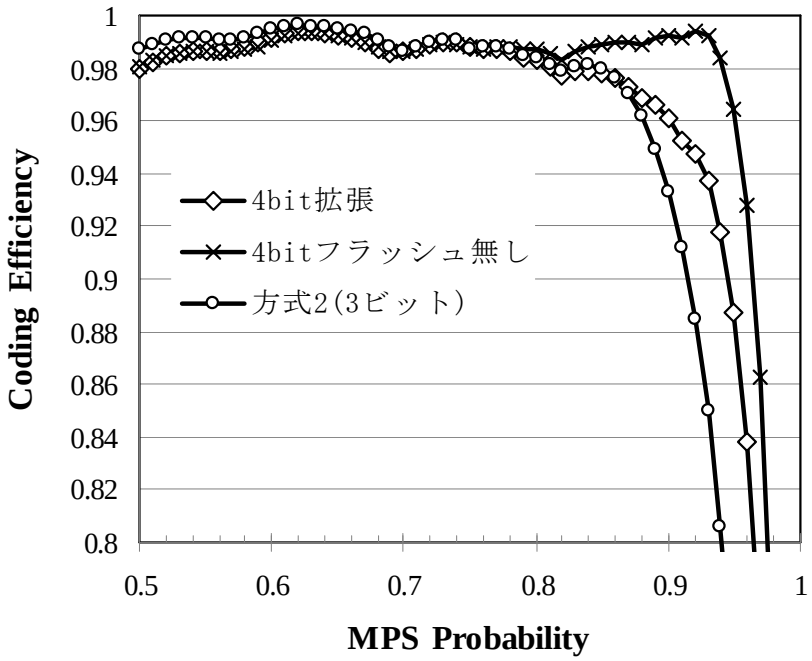


図 3-7　符号化効率(４ビットシステム)

3.4 むすび

算術符号をハフマン符号と同様に状態遷移先と符号とを出力する状態遷移テーブルを参照して符号化処理が実行できるようにすることにより，処理の簡易化を実現する２値算術符号器 STT-coder を提案した．本章ではその基礎検討として，レジスタ長３ビット，４ビットの符号器の設計，性能評価を行った．

レジスタ長３ビットという最小限の精度であっても MPS 確率が低い部分では約 99%の符号化効率を達成しており，アプリケーションによっては充分な性能を有する符号器が実現できることが確認できた．ただし，再正規化を行う場合に必ずフラッシュを行う方式１に比べて，再正規化条件を緩和した方式２の方が高 MPS 確率においては，最大 3%程度高い符号化効率が得られた．今後，高 MPS 確率での符号化効率を高めるためには，レジスタ長を拡張することはもちろん，多少のテーブルサイズの増大は許容し，再正規化条件を緩めて領域幅を広く保つことが符号化効率向上に有効であることを明らかにした．

続いて，低 LPS 確率での符号化効率の向上を図るためレジスタ長を４ビットに増大させる上で，ROM サイズの節約を意図し高 LPS 確率の情報源では 3 ビットレジスタでの精度にとどめることを検討した．しかしこのような考えから汎用的な設計指針を導くのは困難であったため，さらにレジスタ長を増大させる上では 3 ビット設計での方式２に基づきより汎用的な設計方針を確立することを目指すこととする．

第4章　レジスタ長の拡張

4.1　まえがき

第 3 章での 3 ビット STT-coder の評価結果から，方式 1（有効領域幅が最大有効領域の 1/2 以下で常にフラッシュ）では拡大フラッシュを導入しても効率の低下がある程度生じるのに対し，方式 2（最上位ビット確定時には通常の再正規化）では効率の低下が許容範囲におさまると想定できることがわかった．また，レジスタ長が 3 ビットではたとえ通常の算術符号であっても高 MPS 確率の情報源には充分対応できないので実用的な符号の実現にはより長いレジスタ長での設計が必要となる．そこで，方式 2 を前提としてより一般的な方式設計について 6 ビットの例をとりあげ本章で考察することとする [20), 24)].

なお，3 ビットシステムでは再正規化の対象となるサイズとして 1～4 を考えればよく拡大フラッシュの導入により無駄な領域が生じないことも保証されるが，レジスタ長をより長くしたシステムでは，有効領域が最大有効領域幅の 1/2 以下になっても上位ビットが定まらなければ，継続して対象情報源に応じた複数通りの領域分割を行うことになる．以下，STT-coder のレジスタを 6 ビットに拡大し，その領域分割テーブルの設計指針について検討する．

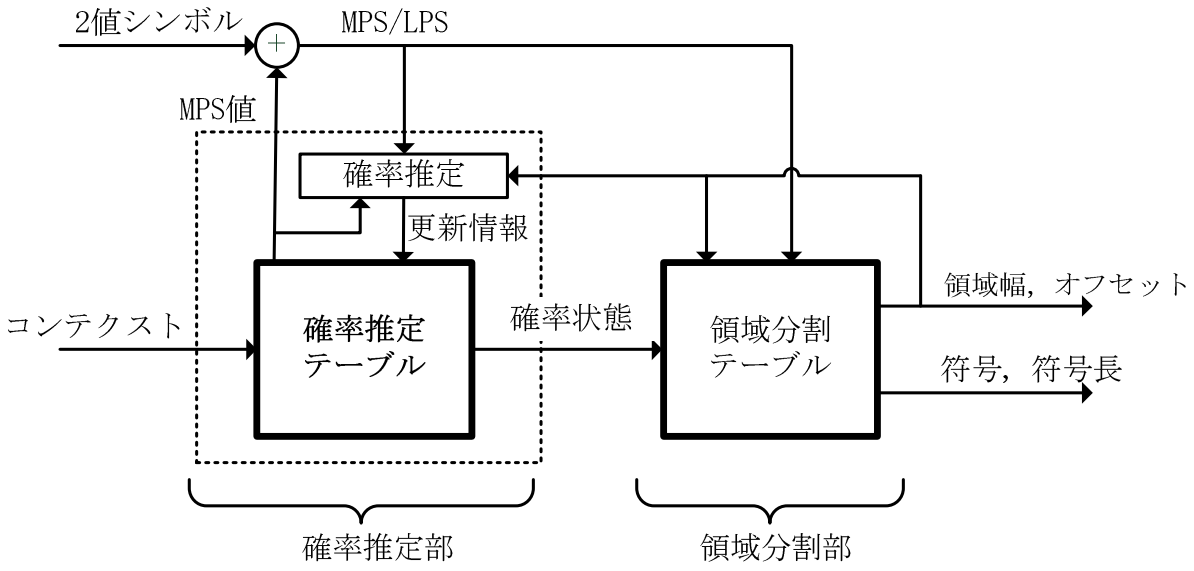


図 4-1　STT-coder ブロック図

4.2　6 ビットシステムへの拡張

4.2.1　6 ビット STT-coder

6 ビットレジスタを有する STT-coder の全体構成を図 4-1 に示す．確率推定部と領域分割部からなり，確率推定部ではシンボルのコンテクストを入力として，そのコンテクストにおける MPS 確率の大きさを示すパラメータである確率状態 S_i を出力する．各確率状態がカバーする MPS 確率範囲は，各状態 S_i を代表する最適 MPS 確率 P_{Mi} に基づき，その範囲内の情報源を符号化した時に，理論的符号化効率が予め定めた設定値 η を超えるように定めている．図 4-2 に設定符号化効率 η=0.99 とした場合の各確率状態（S_0～S_7）に対応する MPS 確率の範囲，及びその符号化効率を示し，表 4-1 に各確率状態の最適 MPS 確率と有効領域幅に対する最適な LPS 幅を示す．ここでは，符号レジスタ長を 6 ビットとしているので，表現できる最大有効領域幅である 64 でも LPS 幅が 1 となる S_7 が MPS 確率最大の状態であり，S_0～S_7 の 8 種類の確率状態が設定される．6 ビットレジスタでは MPS 確率が 0.5 から 0.987 程度の間の情報源に対し 8 つの情報源モデル（確率状態）を設定することで理想的には符号化効率として 99%が期待できることになる．この設計の背景としては確率状態数を 2 の冪乗とすること，3 ビットシス

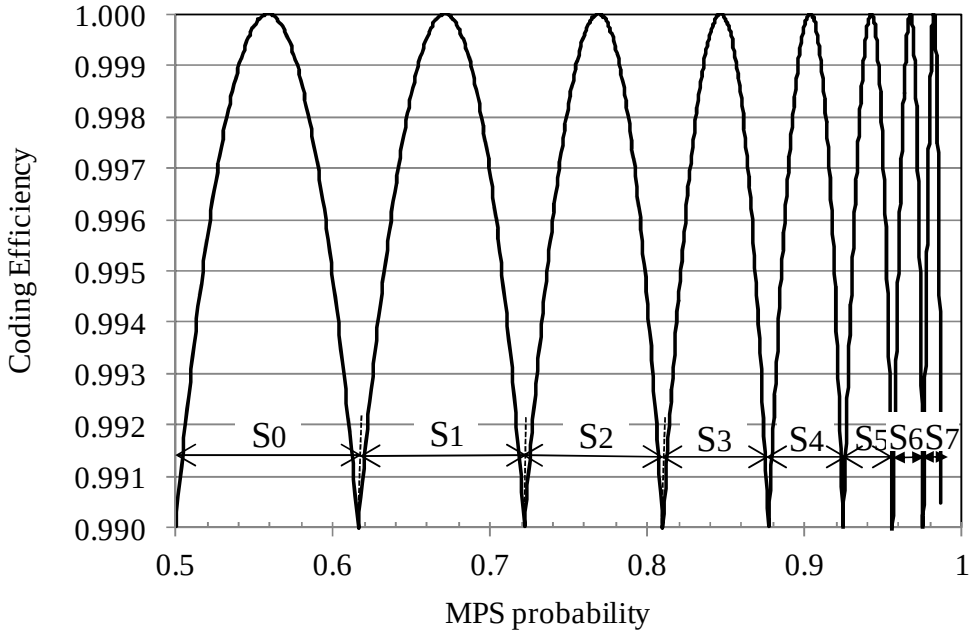


図 4-2　理想符号化効率

表 4-1　各確率状態に対する最適 LPS 幅，最適 MPS 確率

（6 ビットレジスタ，オフセット D=0）

	有効領域幅　A_{ML}										最適 MPS 確率 P_{Mi}
	64	63	62	61	60	…	…	35	34	33	
S_0	28	28	27	27	26	…	…	15	15	15	0.559
S_1	21	21	20	20	20	…	…	12	11	11	0.671
S_2	15	15	14	14	14	…	…	8	8	8	0.769
S_3	10	10	9	9	9	…	…	5	5	5	0.847
S_4	6	6	6	6	6	…	…	3	3	3	0.904
S_5	4	4	4	4	3	…	…	2	2	2	0.942
S_6	2	2	2	2	2	…	…	1	1	1	0.967
S_7	1	1	1	1	1	…	…	1	1	1	0.982

テムでの符号化効率の結果から 99%を目安としたことがあげられる．

領域分割部では，3 ビットレジスタの場合と同様に，出現シンボル（MPS/LPS），確率状態及びその時点での符号化レジスタの有効領域幅に応じて，領域分割テーブルを参照することにより発生符号長，発生符号，次の有効領域幅などを出力する．領域分割テーブルの構成例を表 4-2 に示す．表 4-2 に記載した A_{ML} の種類は $A_{ML}+D$ が 33~64 なので 32 通りであり，D の値の如何によらず 5 ビットで表現できる．

以下，数直線上で現在有効な領域幅を有効領域幅 A_{ML}，そのうち，LPS の領域幅を LPS 幅 A_L，MPS の領域幅を MPS 幅 A_M，過去に出力された符号により定まる数直線下端から最新有効領域下端までの距離をオフセット D と呼ぶ．表 3-1，表 3-2 では入力に領域幅インデックスを導入しているが，これは有効領域の下端と送出済み符号で決まるアドレスであるオフセットとの組み合わせである．通常の算術符号では有効領域の下端を絶対アドレスとして記憶している．一方 STT-coder では C レジスタの保持を不要としており，A レジスタの大きさの範囲の中で有効領域の下端を送信済み符号により定まるアドレス（記憶不要）＋オフセット D で記憶する．このため，この時，オフセットと有効領域との和は，有効領域幅の 1/2 より大きく，有効領域の最大幅を超えない．以下で検討するように D として 0 以外の値を許容することにより LPS 幅設定の自由度が高まる点で符号

化性能の向上効果が期待できるが領域分割テーブルのサイズは増大し，同時に有効領域幅の期待値が低下することに伴う問題も生じる．3 ビット STT-coder ではオフセットの種類が 0 から 3 に限られており，とりうる状態についても個別に検討できるのに対し，n ビット STT-coder の一般的な設計ではオフセットの種類が最大 2^{n-1} 通り考えられ，より汎用的な設計指針が求められる．

4.2.2 領域分割テーブルの設計

領域分割テーブルを設計するにあたり，オフセット D，有効領域幅 A_{ML}，確率状態 S_i の全ての発生パターンに対する LPS 幅 A_L を定める必要がある．領域幅の最大値が 64(=2^6)なので，各パラメータは

$$64/2 < A_{ML} + D \leq 64 \tag{4-1}$$

$$A_{ML}/2 \leq A_M < A_{ML} \tag{4-2}$$

の範囲の値をとる． 有効領域幅 A_{ML} としては上記範囲を満たす全ての値を用意する．これは，高 MPS 確率の確率状態では LPS 幅 A_L=1 を選択することが必要となるためである．確率状態は 4.2.1 で述べたように 8 種類であり，オフセットの種類数 Nについては，4.2.3 で最適化を検討する．

LPS 幅 A_Lについては，各確率状態と有効領域幅に基づき，表 4-1 のように各確率状態の最適 MPS 確率に最も近い比にしたがって領域分割を行うのが個別には理想であるが，ある LPS 幅の選択が許容されるためには，有効領域内で LPS を上位あるいは下位のいずれかに配置した時，次の入力シンボル発生後のオフセットが必ず許容されるオフセット値の中にあることが条件となる．なお， LPS を上位あるいは下位のどちらにも配置することが可能な場合には，次の入力シンボル発生後の領域幅の最小値（MPS あるいは LPS 選択後の領域幅の小さい方）がより大きい方の配置を優先して選択する．

表 4-2　6 ビット STT-coder の領域分割テーブル

入力				出力			
3bit	5bit	2bit	1bit	6bit	3bit	5bit	2bit
S_i	A_{ML}	D	MPS /LPS	符号	符号長	A_{ML}	D
S_0	33	0	M	-	0	17	16
			L	00	2	64	0
	34		M	-	0	18	16
			L	00	2	64	0
	35		M	-	0	19	16
			L	00	2	64	0
	...		M	-	0	20	16
			...	...	...	...	...
...	...		...	...	...	...	...
			...	...	...	...	...
	...		...	...	...	...	...
			...	...	...	...	...
	...		...	...	...	...	...
			...	...	...	...	...
S_7	...		...	...	...	...	...
			...	...	...	...	...
	63		M	-	0	62	0
			L	111110	6	64	0
	64		M	-	0	63	0
			L	111111	6	64	0
S_0	17	16	M	-	0	9	24
			L	010	3	64	0
	18		M	-	0	10	24
			L	010	3	64	0
	...		...	...	...	...	...
			...	...	...	...	...
...	...		...	...	...	...	...
			...	...	...	...	...
...	...	...	...	...	...	...	...
			...	...	...	...	...
S_7	...	28	...	...	...	...	...
			...	...	...	...	...
	35		M	-	0	34	28
			L	111110	6	64	0
	36		M	-	0	35	28
			L	111111	6	64	0

4.2.3　オフセット種類数 N と符号化性能

符号器の簡易化のためにはオフセットの種類を制限することが望ましいが，少なすぎると確率状態と有効領域幅に対する適切な LPS 幅が設定できないために符号化効率に悪影響を及ぼす．そこで，オフセットの種類数 N と符号化性能との関係を検討することとした．

最初に，オフセットを 1 種類（N=1，許容オフセット D=0）とした場合の符号化効率を調べた．図 4-3 に MPS 確率が 0.5～1.0 の範囲で 0.0025 おきに均一に分布するマルチコンテクストの 2 値シンボル系列を対象とし，シミュレーションにより算出した MPS 確率対符号化効率の結果を示す．オフセットが D=0 のみでは，LPS 幅の選択肢が極めて制限されるため，特に低 MPS 確率の確率状態における符号化効率が低下している．ただし，オフセットが 0 であるため有効領域幅 A_{ML} の期待値は大きいことと，A_L=1 は常に確保されているため，高 MPS 確率の確率状態における符号化性能は極めて高い．

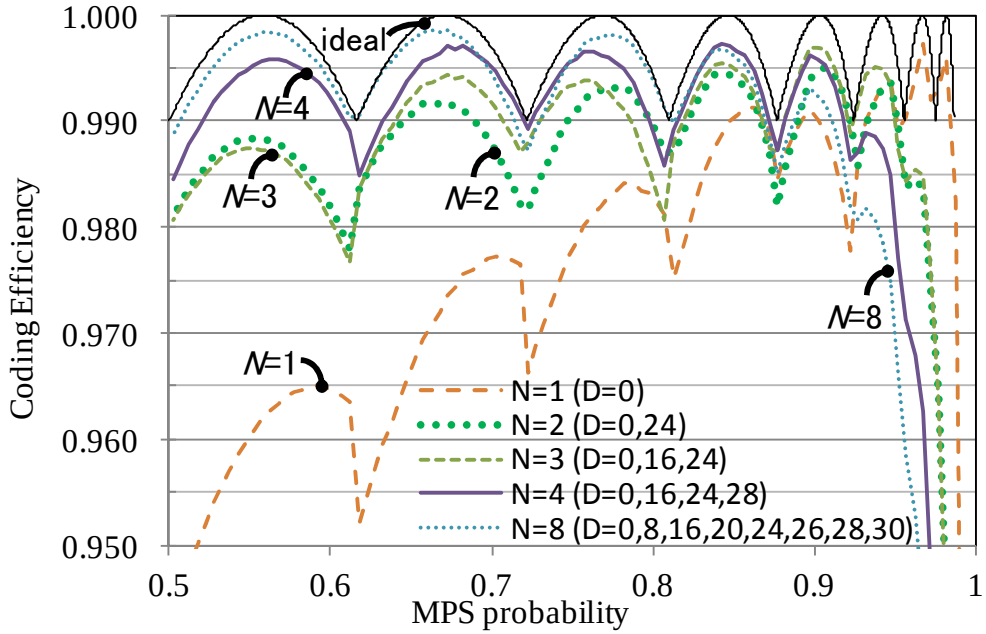


図 4-3　符号化効率（6 ビット STT-coder）

A_M \ A_{ML}	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
63																																1
62																															1	2
61																														1	2	-
60																													1	2	-	4
59																												1	2	-	4	-
58																											1	2	-	4	-	6
57																										1	2	3	4	-	6	-
56																									1	2	3	-	-	6	-	-
55																								1	2	3	-	-	6	-	-	-
54																							1	2	3	-	-	6	-	-	-	10
53																						1	2	3	-	-	6	-	-	-	10	-
52																					1	2	3	-	5	6	-	-	9	10	-	-
51																				1	2	3	-	5	-	-	-	-	-	-	-	-
50																			1	2	3	-	5	-	-	8	9	10	-	-	-	14
49																		1	2	3	-	5	-	-	-	-	-	-	-	-	-	-
48																	1	2	3	-	5	-	-	-	9	-	-	-	-	14	15	-
47																1	2	3	-	5	-	-	**8**	9	-	-	-	-	14	-	-	-
46															1	2	3	-	5	-	-	**8**	-	-	-	-	-	14	-	-	-	-
45														1	2	3	-	5	-	-	8	-	-	-	-	13	14	-	-	-	-	-
44													1	2	3	-	5	-	-	**8**	-	-	-	12	13	-	-	-	-	-	-	-
43												1	2	3	-	5	-	-	**8**	-	-	-	-	-	-	-	-	-	-	-	-	-
42											1	-	3	4	5	-	-	8	-	-	-	12	13	-	-	-	-	-	-	**20**	21	22
41										1	-	3	4	-	-	7	**8**	-	-	-	-	-	-	-	-	-	-	-	**20**	-	-	-
40									1	2	3	4	-	-	7	-	-	10	11	12	13	-	-	-	-	-	19	**20**	-	-	-	-
39								1	2	-	4	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
38							1	2	-	4	-	-	7	-	-	-	-	-	-	-	-	-	-	-	-	**20**	-	-	-	-	-	-
37						1	2	-	4	-	-	7	-	-	-	11	-	-	-	-	-	-	-	-	**20**	-	-	-	-	-	-	-
36					1	2	-	4	-	6	7	-	-	10	11	-	13	-	-	16	17	18	19	**20**	-	-	-	-	-	-	-	28
35				1	2	-	4	-	-	-	-	-	-	-	-	-	-	-	**16**	-	-	-	-	-	-	-	-	**26**	-	**28**	-
34			1	2	-	4	-	6	7	-	9	10	11	-	-	-	-	**16**	-	-	-	-	-	-	-	-	-	**26**	-	**28**	-	-
33		1	2	3	4	-	6	-	-	-	-	-	-	-	-	-	**16**	-	-	-	-	-	-	-	-	-	**26**	-	-	-	-	-
32	1	2	3	-	5	6	-	-	9	10	-	-	-	-	15	**16**	-	-	-	-	-	-	-	**24**	25	**26**	-	-	-	-	-	-
31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**24**	-	-	-	-	-	-	-		
30	3	4	5	6	-	8	9	10	-	-	-	14	15	16	-	-	-	-	-	22	23	**24**	-	-	-	-	-	-				
29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-						
28	5	6	-	**8**	9	-	-	-	13	14	15	-	-	-	-	-	21	22	23	-	-	-	-	-								
27	-	-	**8**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-										
26	-	**8**	-	-	-	12	13	14	-	-	-	-	-	**20**	21	22	-	-	-	-												
25	**8**	-	-	-	-	-	-	-	-	-	-	-	**20**	-	-	-	-	-														
24	9	10	11	12	13	-	-	**16**	17	18	19	20	-	-	-	-																
23	-	-	-	-	-	-	**16**	-	-	-	-	-	-	-																		
22	-	-	-	-	-	**16**	-	-	-	-	-	-																				
21	-	-	-	-	**16**	-	-	-	-	-																						
20	-	-	15	**16**	-	-	-	-																								
19	-	-	-	-	-	-																										
18	-	**16**	-	-																												
17	**16**	-																														

図 4-4　選択された MPS 幅（6 ビット STT-coder, D=0）

次に N=2 とし，D=0 に追加すべき最適なオフセット値を同様にマルチコンテクストの 2 値シンボル系列での平均符号化効率により求めると D=24 であった．このとき LPS 幅の選択肢が広がり，低 MPS 確率の確率状態での効率は大きく向上しているが，オフセット D=24 が加わることにより有効領域幅の小さいケースが増えるため高 MPS 確率の確率状態での効率は下がっている．同様に N=3 については，D=0,24 に追加するオフセット値としては D=16 が良いとわかった．さらに，N=4 について検討すると，D=28 の追加による低 MPS 確率の確率状態での効率向上への寄与が大きく，N=4 での最適なオフセット値の組み合わせは(D=0, 16,24,28)と判断した．

なお，さらに Nを増大させても，低 MPS 確率での確率状態での効率向上は僅かであるのに対し（図 4-3 に N=8 の例も示す），高 MPS 確率の確率状態での効率低下が大きく，全体としての平均的性能は低下するため，オフセットの種類数は N=4（D=0,16,24,28）を最適とみなすこととした．

図 4-4 には，オフセット D=0 において，各有効領域幅 A_{ML}に対して選択された MPS 幅 A_M（＝A_{ML}－A_L）を示す．有効領域幅 A_{ML}と MPS 幅 A_Mのマトリクス内で，数値が示された箇所はその値が MPS 幅として選択されたことを示す．その中で，背景がグレーの箇所は LPS が有効領域内で上方に配置されていることを表している．

4.2.4　LPS 幅修正による有効領域の拡大効果

N=4 では D=28 が追加され，低 MPS 確率での確率状態での効率向上が実現できたが，D が大きいほど有効領域幅の期待値は小さくなる．そこで D=28 の状態からなるべく有効領域幅の大きい状態に遷移できるよう個別符号化での領域分割比の最適化を若干犠牲にして有効領域の増大化を優先する処理を検討した．例えば，図 4-5(b)に示すように，D=28 で A_L=4（LPS 下方配置）では MPS/LPS のいずれが発生しても D=0 となり次の A_{ML}として大きな値が期待できる．そこで図 4-5 (a)のように，A_L=4 の前後の値である A_L=3 もしくは 5 を禁止し，これらの場合はすべて A_L=4(LPS 下方配置)に修正することを考えた．またオフセット D=24 では，A_L=8（LPS 下方配置）が同様の効果を有する．そこで D=24 では A_L=8 の前後の値である A_L=7 もしくは 9 を禁止し，すべて A_L=8（LPS 下方配置）への変更を試みた．図 4-6 に N=4 (D=0,16,24,28)の場合に，上記両オフセットについて LPS 幅修正を施した場合の符号化効率を示す．LPS 幅修正に伴う悪影響が大きいのは確率状態 S_2 の付近であるため，その近傍では僅かに符号化性能の低下が

認められるが，高 MPS 確率での効率および平均的な効率は向上しており，所望の効果が表れているといえる．

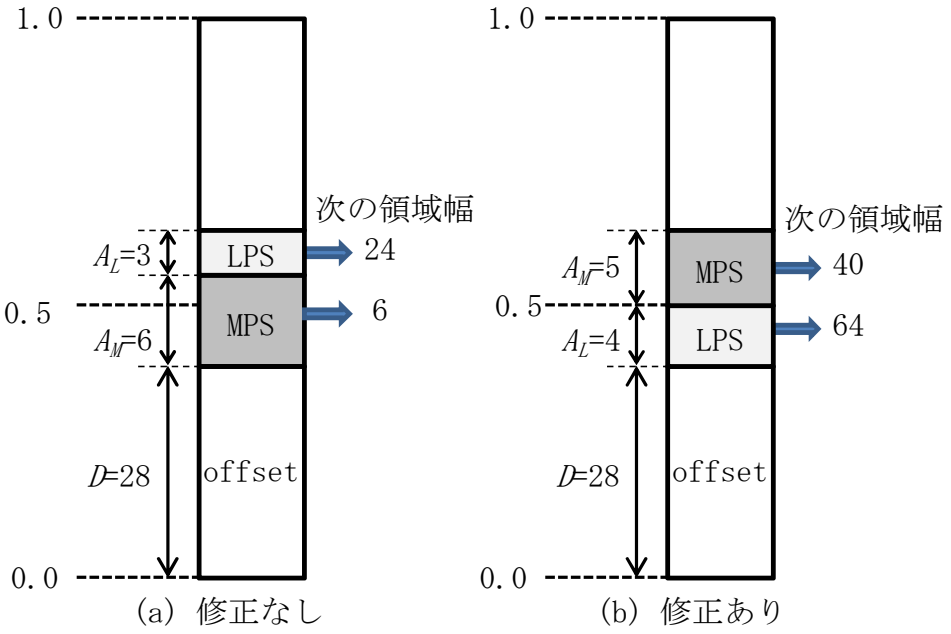


図 4-5　LPS 幅の修正

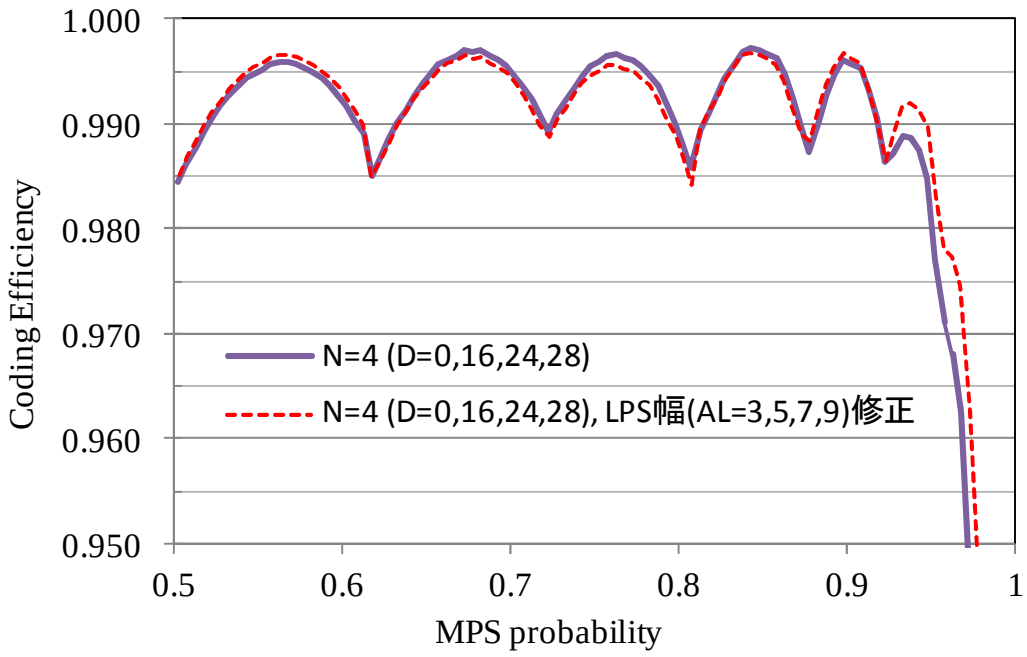


図 4-6　符号化効率（6 ビット STT-coder）

4.3 まとめ

本章では，レジスタ長 3 ビットの STT-coder の評価に基づき，高 MPS 確率の情報源に対応するため，レジスタ長を 6 ビットに拡張するにあたり，最上位ビットの確定時には通常の再正規化を行う簡易化案（方式 2）を基本とし，設計パラメータであるオフセット数 N に着目して，その種類と値の最適化について検討した．その結果，領域分割テーブルのサイズは N と共に大きくなるのに対し，符号化効率については N が大きいほど低 MPS 確率の確率状態での符号化効率を上げることができるものの，高 MPS 確率の確率状態での効率は逆に低下することを明らかにした．この結果，6 ビットシステムでは N=4 が平均的符号化効率の点でほぼ最適とみなせ，その時のオフセットの組み合わせは(D=0,16,24,28)が良いこと，さらにオフセットの大きな D=28,24 においては特定の LPS 幅については現時点での符号化の最適化よりも将来の有効領域の増大化を優先させることにより，広範囲の確率状態に対しより良好な符号化性能の実現が可能であることを示した．

6 ビット STT-coder は MPS 確率が 0.5 から 0.95 の範囲の情報源に対し 98.5% 以上の符号化効率を実現することがわかった．なお，より高い MPS 確率を有する情報源に対してはより長いレジスタ長でのシステム設計が必要になる．今回の検討結果は，より長いレジスタ長におけるシステムを設計する上でも充分汎用的な設計指針を与えるものと考えている．なお，本章では，シンボル出現確率が既知の情報源を想定した静的な符号化特性を検証した．次のステップとして次章では，動的な確率推定を組み込んだより実用的な符号器設計を検討する．

第5章　確率推定

5.1　まえがき

前章までは、STT-coder における領域分割部の処理について検討し、レジスタ長 6 ビット、確率状態 8 状態で実用レベルのテーブル参照型算術符号が実現できることを示した。ただしそれは、静的な情報源シンボルを想定した場合のものであり、統計的な性質が未知あるいは非定常な情報源への動的な対処は考慮していない。本章では図 4-1 に示した STT-coder の確率推定部において，統計的な性質が未知あるいは非定常な情報源に対し，情報源シンボルの生起確率を動的に学習し，生起確率に適した符号化パラメータを選択する確率推定とその動的符号化特性について考察を行う．

エントロピ符号化における確率推定方法としてベイズ確率推定(Bayesian Probability Estimation)，及びそれを簡易化した状態遷移型確率推定(State Transition Probability Estimation)が広く利用されている．各情報源シンボルの出現度数を計数し，全情報源シンボルの出現度数に対する比率を各シンボルの出現確率とするベイズ確率推定は，そのカウンタ精度でカバーしうる確率範囲に対しては非常に高い性能を有する．ただし，シンボルの種類数分のカウンタが必要であることや，カウント値から出現確率を求めるために出現シンボルごとに常に推定確率の見直しが必要でかつ除算を伴うため，高速化を図る上での障害となっていた．そこでより実現の簡易な方法として，予め設定された状態間をある一定の規則に従って遷移し，滞在状態から確率推定を行なう状態遷移型確率推定も検討されてきた．状態遷移型確率推定に関する過去の報告の代表的なものとして Q-Coder[8]がある．Q-Coder は推定確率を定めた状態の遷移を再正規化のタイミングで行っており、QM-coder[12),13)]や MQ-coder[2),9)]にもその考えが受け継がれている．しかしながら動的性能の設計に関する明確な方針の説明はなく、確率状態の更新法則を定めた上でその動的符号化性能の評価から各状態を設計したとも考えられる．また MELCODE では MPS のカウンタを基にした動的設計が行われている[25)]．文献 25)では LPS 確率が高い部分で符号化効率が低下することから LPS 確率が高い領域ほど状態を密にとる方針が示されている．

また状態遷移型確率推定（MPS カウンタ利用）とベイズ型確率推定（MPS/LPS カウンタ利用）を比較し，LPS 確率の高い部分では後者を採用し，両者を組み合わせることで符号化性能と装置規模のバランスのよい確率推定を実現する状態遷

移・ベイズ推定切替型確率推定が筆者らにより提案されている[26),26),27)].

本章ではベイズ推定型、状態遷移型、および切替型確率推定の確率推定方式としての性能を明らかにした上で画像符号化への適用結果を述べ，それらの特徴を考察する．次に，STT-coder に適した簡易な確率推定として状態遷移型推定を取り上げ，LPS 確率が高い部分で動的符号化性能が低下するという従来からの問題点に関する汎用的考察に基づき、その解決策を検討する．

5.2 各種確率推定の性能

5.2.1 ベイズ型確率推定と状態遷移型確率推定

ベイズ確率推定においては LPS(Less Probable Symbol：劣勢シンボル)と両シンボルの累積出現度数をカウンタ N_L と N_T を用いて 2 値シンボルが出現する度に計数する．カウンタは，N_L=1，N_T=2 を初期値とし，N_T がある最大値 N_{MAX} を越えた場合には， N_L と N_T の値を⌈N_L/2⌉, ⌈N_T/2⌉（⌈・⌉: ceiling 関数）に半減する．また，N_L と N_T の他に，MPS(More Probable Symbol：優勢シンボル)の値(0 または 1)を記憶する 1 ビットのメモリを用意しておき，カウンタが $N_T < N_L \times 2$ の条件を満たした場合には，MPS の値を反転し，N_L を $N_T - N_L$ とする．その結果，N_L の値は常に N_T の 1/2 以下となるので，N_T の最大値を 255 とした場合，N_T, N_L と MPS 値の記憶にそれぞれ，8 ビット，7 ビット，1 ビットを要し，情報源当たり計 16 ビットのメモリを必要とする．

状態遷移型確率推定では，予め推定 LPS 確率が定められた複数の状態間を，ある一定の遷移規則に従って遷移し，符号化時点での滞在状態における推定確率を符号化に利用する．ある状態から他の状態への遷移は，滞在状態で生じた情報のみにより起こり，過去の状態における出現度数は参照しないものとする．

本項では，各状態で発生した MPS の発生度数のみに注目して状態間を遷移する状態遷移規則を例として取り上げる．この遷移規則では，算術型 MELCODE[15),16)]を前提としており表 5-1 に示す 29 状態を設定し，各状態毎に A-value と呼ばれる LPS 確率に相当するパラメータと MPS 最大出現度数の既定値を定めておく．符号化時には，その時点での滞在状態の A-value をもとに算術型 MELCODE で符号化を行う．この時，滞在状態での MPS 出現度数を計数し，MPS の出現度数がその状態での既定最大値に達したならば，一段上の状態(より推定 LPS 確率の低い状態)に遷移し，MPS カウンタを 0 にリセットする．逆に，MPS カウンタが規定値に達する前に，LPS が発生した場合には，一段下の状態

に遷移し，MPS カウンタを 0 にリセットする．表 5-1 の場合，MPS カウンタに 11 ビット，状態のインデックスを記憶するのに 5 ビット，MPS の値を記憶するのに 1 ビット，計 17 ビットのメモリが必要となる．

5.2.2 ベイズ確率推定と状態遷移型確率の比較

図 5-1 に両確率推定方式を使った算術型 MELCODE での符号化効率を示す．ここで，符号化効率は情報量を 1 シンボルあたりの符号長で除したものである．符号化効率は，計算機で発生させた LPS 確率が $1/2^{-n/4}$(n=4～52)で独立同一分布の擬似ランダム系列をそれぞれ実際に符号化して算出した．試行したシンボル数は各系列 100 万個である．ベイズ推定の符号化効率は，LPS 確率が 1/2 近辺では非常に高いが，LPS 確率が低くなるにつれ，カウンタ精度の制限により次第に低下する．また，カウンタ精度が低い(N_{MAX}の値が小さい)と精度の高い場合に比べてより早く効率の降下が始まる．一方，状態遷移型の場合には，幅広い LPS 確率をとる情報源に対してある程度の性能が得られるが，LPS 確率が 1/2 近くでは，ベイズ推定に比べて符号化効率が若干劣っている．

ベイズ推定と状態遷移型との違いの一つは，推定確率が更新される時間間隔にある．つまり，ベイズ推定では，1 シンボル毎に推定確率が更新されるのに対し，状態遷移型では多くの場合(MPS 発生時)，複数のシンボルの観察後に推定確率が更新される．このため，ベイズ推定では，カウンタ精度でカバーできる確率範囲では，推定精度が高くなり，状態遷移型に比べて優れた符号化効率が得られる．一方，推定確率の更新頻度の少ない状態遷移型では，同じメモリ量を要するベイズ推定に比べ，より広範囲の確率までカバーでき，低 LPS 確率についてより高い符号化効率が得られる．

表 5-1　状態遷移型確率推定の A-value と MPS 最大出現度数

状態	A-value	MPS 最大出現度数	状態	A-value	MPS 最大出現度数
S_0	1/2	1	S_{15}	5/128	14
S_1	7/16	1	S_{16}	1/32	16
S_2	3/8	1	S_{17}	3/128	24
S_3	5/16	1	S_{18}	1/64	32
S_4	1/4	2	S_{19}	3/256	48
S_5	7/32	2	S_{20}	1/128	64
S_6	3/16	3	S_{21}	3/512	96
S_7	5/32	3	S_{22}	1/256	128
S_8	1/8	4	S_{23}	3/1024	192
S_9	7/64	5	S_{24}	1/512	256
S_{10}	3/32	6	S_{25}	1/1024	512
S_{11}	5/64	7	S_{26}	1/2048	1024
S_{12}	1/16	8	S_{27}	1/4096	2048
S_{13}	7/128	10	S_{28}	1/8192	---
S_{14}	3/64	12			

5.2.3　状態遷移・ベイズ推定切替型確率推定

（１）確率推定アルゴリズム

状態遷移・ベイズ推定切替型確率推定 [26),27),28)]では，上記２つの確率推定方式の性能を向上させるべく，シンボルの推定出現確率に応じて両者を切り替える，つまり LPS 確率が高い場合にはベイズ確率推定を，LPS 確率が低い場合には状態遷移型確率推定を用いる方式である．両推定方式の切り替えは以下のように行う．初期推定方式はベイズ推定とし，カウンタが $N_L < R \times N_T$ の条件を満たした場合には，図 5-2 に示すように確率推定を状態遷移型に切り替える．ここで R は，確率推定方式が状態遷移型に切り替わる境界確率である．この境界確率は，図 5-1 に示したベイズ推定と状態遷移型の符号化効率の特性が交差する確率に設定している．また，状態遷移型確率推定の初期状態は，境界確率 R に最も近い推定確率に対応する状態 S_b とする．状態遷移型確率推定では，S_b から S_{28} の間で遷移を行い，状態 S_b で LPS が出現した場合には，確率推定方式をベイズ確率推定に再度

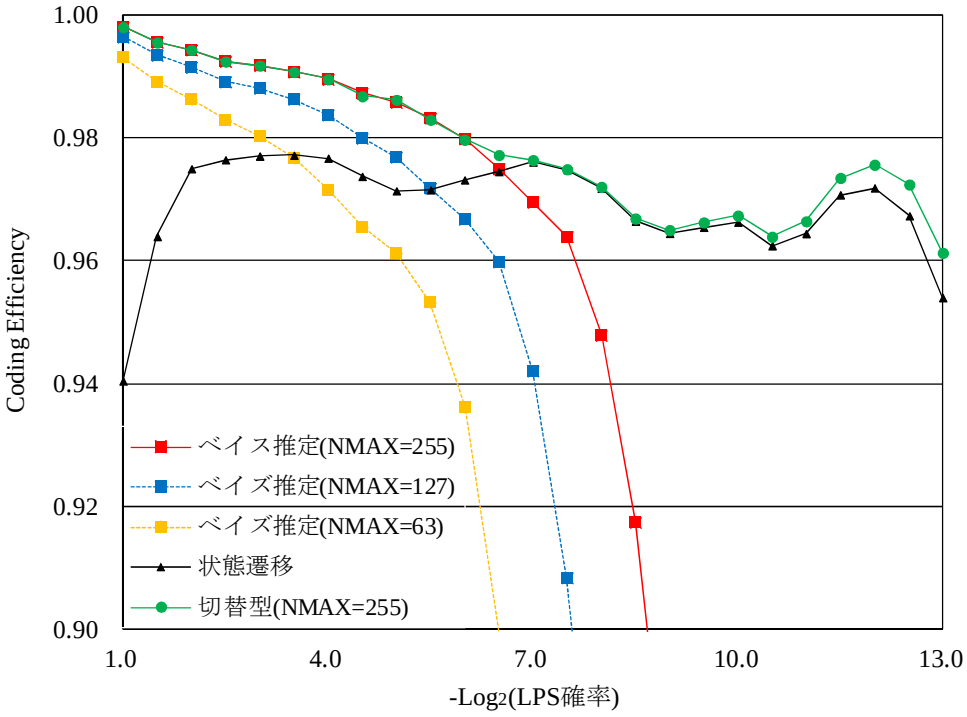


図 5-1　擬似ランダム系列に対する符号化効率

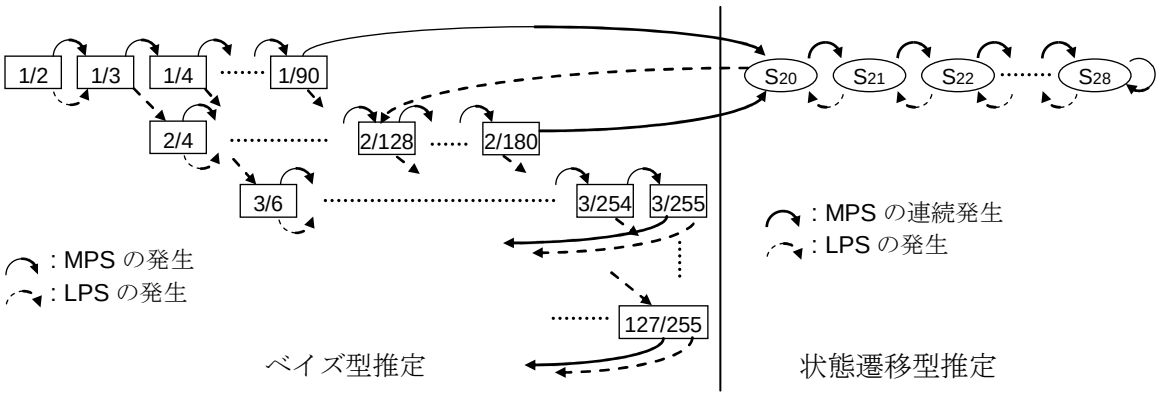


図 5-2 状態遷移・ベイズ推定切替型確率推定(N_{MAX}=255)

表 5-2 確率推定方式切替条件

	ベイズ確率推定でのカウンタ最大値:N_{MAX}			
	255	127	63	31
R	1/90	1/45	1/12	1/4
S_B	S_{20}	S_{18}	S_{12}	S_5
N_{L0}/N_{T0}	2/128	2/64	3/32	5/16

切り替える.確率推定方式がベイズ確率推定に切り替わった直後のカウンタ値は,状態 S_{b-1}での推定確率に相当する N_{L0}, N_{T0}にリセットされる(境界確率 R等の値は表 5-2 参照).

以下では,ベイズ推定でのカウンタ最大値 N_{MAX}をまず 255 として議論をすすめる.これは,表 5-1 の状態遷移型確率推定に要するメモリ量以下でのカウンタ最大値であり,図 5-1 に示したベイズ推定の静的符号化特性の中では最も優れた符号化効率が得られるからである.切替方式では,どちらの確率推定方式を使用しているかを記憶する 1 ビットのフラグを要するが,両確率推定方式で使用するカウンタに必要なメモリは共有でき,また状態遷移型確率推定での状態数は減少し状態インデックスは 4 ビットで済むため,必要なメモリはやはり計 17 ビットとなる.

図 5-1 に擬似ランダム系列に対する状態遷移・ベイズ推定切替型確率推定

(N_{MAX}=255)の符号化効率を示した．状態遷移型とベイズ推定の確率推定方式の特性がスムーズに切り替わっており，従来方式に比べ，ほぼ同一のメモリ量でより高い符号化効率が得られていることがわかる．

（２）画像データ符号化結果

切替型確率推定，ベイズ確率推定，状態遷移型確率推定を算術型 MELCODE に適用し，多値画像と２値画像を符号化した際の圧縮率(原画サイズ／符号量)を表 5-3(N_{MAX}=255)に示す．表 5-3 の多値画像は，自然画像(Cafe，Woman:SCID 画像，８bits/sample)と自然画／文書混在画像(Compound1.r，Compound2.r：８bits/sample)であり，これらを多値静止画像符号化標準 JPEG-LS part-2[29]の符号化モデルを利用して符号化した．JPEG-LS part-2 では，多値である予測順位情報を複数コンテクスト（コンテクスト数は 880）の２値シンボル系列に変換し，２値の算術符号化を適用している．以下では算術符号として算術型 MELCODE を用い，ベイズ確率推定方式，状態遷移型確率推定方式，切替型確率推定方式の３方式を検証した．JPEG-LS part-2 では，発生する２値シンボルの多くが 1/2 に近い出現確率となるため，切替型確率推定方式による圧縮率は，ベイズ確率推定方式の値にかなり近いが，いずれの従来方式よりも高い符号化効率が得られている．

表 5-3 の後半５枚の画像は，CCITT ファクシミリテストチャート(＃2，＃4，＃6)と SCID 多値自然画像を誤差拡散法により２値化した２値中間調画像(Cafe，Woman)であり，いずれも２値画像であるので２値画像符号化の国際標準である JBIG[12]の符号化モデル（sequential 方式，コンテクスト数は 1024）を利用して符号化した．ただし，本稿では算術符号化部を算術型 MELCODE におきかえ，同様にベイズ確率推定方式，状態遷移型確率推定方式，切替型確率推定方式を検証した．文書画像の場合は，自然画像の場合とは逆に，LPS 確率が非常に低いコンテクストが占める比率が高いため，ベイズ推定の圧縮率は状態遷移型に比べて大きく劣るが，切替型は状態遷移型と同程度の圧縮性能を有している．中間調画像では，LPS 確率の高いコンテクストが文書画像に比べて多く発生するので，切替型の圧縮率はベイズ推定とほぼ同程度であり，状態遷移型と比べて若干高い圧縮率を示す．

表 5-3　画像データの圧縮率

	画像	ベイズ推定	(diff %)	状態遷移型推定	(diff %)	状態遷移・ベイズ推定切替型推定							
						NMAX=255	NMAX=127	(diff %)	NMAX=63	(diff %)	NMAX=31	(diff %)	
多値画像	Café	1.604	(0.00)	1.526	(5.14)	1.604	1.601	(0.16)	1.597	(0.47)	1.588	(1.02)	
	Woman	1.831	(0.00)	1.746	(4.89)	1.831	1.829	(0.15)	1.823	(0.44)	1.813	(0.99)	
	Compound1.r	6.729	(0.26)	6.523	(3.43)	6.747	6.763	(-0.25)	6.762	(-0.23)	6.741	(0.09)	
	Compound2.r	6.250	(0.19)	6.003	(4.29)	6.261	6.259	(0.03)	6.248	(0.22)	6.222	(0.62)	
2値画像	#2	43.394	(38.82)	60.300	(-0.11)	60.237	60.585	(-0.58)	60.707	(-0.78)	60.614	(-0.62)	
	#4	8.821	(3.34)	9.547	(-4.52)	9.115	9.256	(-1.52)	9.426	(-3.29)	9.549	(-4.54)	
	#6	32.558	(25.22)	40.913	(-0.35)	40.770	41.084	(-0.76)	41.156	(-0.94)	41.100	(-0.80)	
	Café	1.384	(0.00)	1.370	(1.08)	1.384	1.391	(-0.50)	1.392	(-0.53)	1.384	(0.04)	
	Woman	1.437	(0.00)	1.423	(0.94)	1.437	1.445	(-0.56)	1.445	(-0.59)	1.438	(-0.08)	

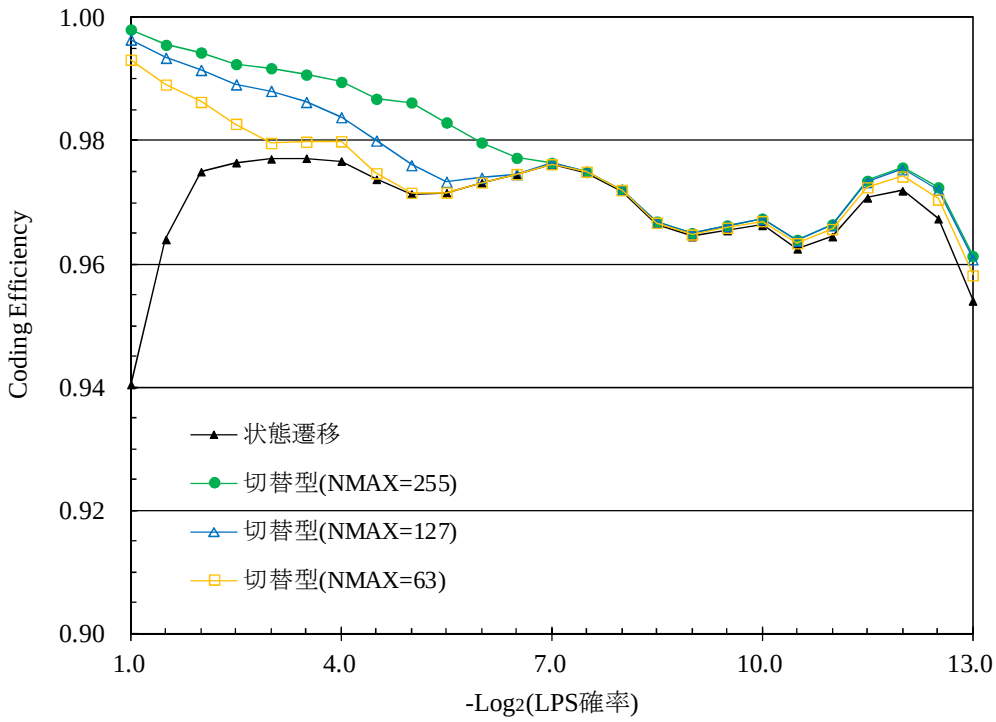


図 5-3　擬似ランダム系列に対する符号化効率

（３）確率変動に対する追随性の検証

切替型確率推定の圧縮性能は，ほとんどの画像について，いずれの従来方式と比べても同等以上であるが，CCITT テストチャート＃4 のような文書画像では，状態遷移型に比べて僅かに劣っている．一般に，情報源シンボルの出現確率が変動する場合の追随性能は，状態遷移型確率推定の方がベイズ確率推定より，特に LPS 確率が上昇する場合において優れていると考えられる．そのため，確率変動の激しい情報源に対しては，比較的高い LPS 確率においてもベイズ確率推定より状態遷移型確率推定の方が優ることがあり，テストチャート＃4 がこの場合に相当すると推定される．

そこで，この問題に対処するため，ベイズ確率推定におけるカウント最大値 N_{MAX} の値を低減し，より高速な追随性を実現することの効果を調べた．N_{MAX} の値を低くすることにより静的な情報源に対する符号化効率は低下するが，カウンタの半減処理がより頻繁に行なわれることと状態遷移型推定の状態が占める割合が増えることにより確率の変動に対する追随性が向上すると考えられるからである．

図 5-3 に切替型で N_{MAX}=127，63 とした際の静的符号化特性を N_{MAX}=255 の場合と共に示した．なお，N_{MAX} の値を低減すると状態遷移での状態数は増加するが，総メモリ量は最大でも 18 ビットでおさまる．LPS 確率が 1/2 近くでは，N_{MAX} の値が低くなるにつれて静的符号化効率は低下し，状態遷移型がより多く使われていることがわかる．次に，実画像に対する圧縮率を測定し，表 5-3 に示した(N_{MAX}=127，63，31)．自然画像に対する圧縮率は，N_{MAX} の値が低くなるにつれて，僅かに低下しているが，その低下は N_{MAX}=255 に比べて，N_{MAX}=63 では 0.5%程度に留まっている．一方，２値画像に対する圧縮率の向上は，多値画像での圧縮率低下を大きく上回っている．これら静的情報源に対する推定精度と動的情報源に対する追随性のトレードオフを考慮すると，推定 LPS 確率 1/64 程度(N_{MAX}=63)での両推定方式の切り替えが最適な圧縮性能を与えるといえる．

5.3 STT-coder における確率推定

5.3.1 状態遷移型確率推定の適用

STT-coder における確率推定を考えるにあたっては，簡易化に適した状態遷移型の確率推定を用いることとする．ここで，従来の状態遷移型確率推定において共通しているのはいずれの方式でも LPS 確率が高い部分で動的符号化性能が低

下することであり、本論文ではこの問題に関する汎用的考察に基づき、その解決策を検討する.

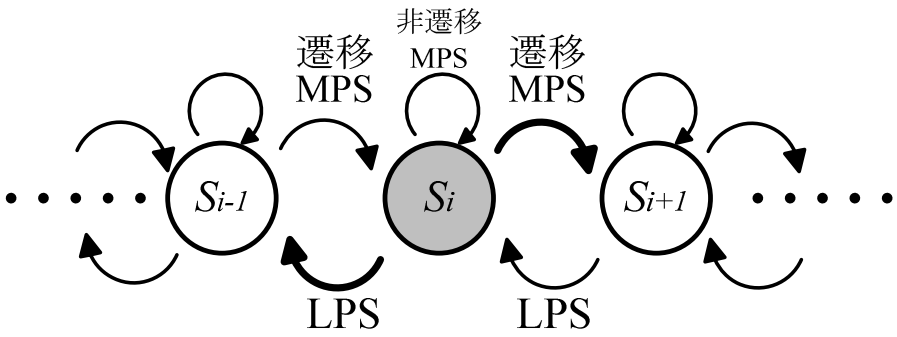


図 5-4　従来型の確率状態遷移

状態遷移型確率推定では, 図 5-4 に示すように LPS の発生で MPS 確率推定を 1 段階下げ（下降), MPS についてはある割合で MPS を選別し, その MPS（遷移 MPS）が発生した場合のみ MPS 確率推定を 1 段階上げる（上昇）方式が従来からとられている. 後者の遷移 MPS 判定については, 前章で述べた MPS カウンタを有する方法 [6] や Q-coder[3] のように再正規化が起きるときのみ遷移 MPS と判断する方法が知られている. MPS カウンタを持つ方法はより高い精度を有すると考えられるが MPS カウンタの値に応じた追加状態を有していると考えるべきであるので, ハードウェア規模が増大する. そこで以下では MPS カウンタは持たないという前提で考察を行うものとする.

まず STT-coder における遷移 MPS の判定であるが, マルチコンテクストモデル情報源として MPS 確率が 0.5 から 0.998 までの範囲で 0.002 おきに均一に分布する情報源を想定し, STT-coder におけるすべての有効領域幅の各発生頻度を求める. 有効領域幅と発生頻度との関係をもとに, ある範囲の有効領域幅ならば非遷移 MPS に, それ以外ならば遷移 MPS と判定して状態遷移（上昇）を行う.

ある状態で最も効率的に符号化される情報源の MPS 確率を最適 MPS 確率と呼ぶ. 次に, 状態 S_i での遷移 MPS 確率（MPS が遷移 MPS とされる割合）を B_i とし, 最適 MPS 確率を p_i, その時の LPS 確率を q_i (＝1-p_i)とすると状態 S_i で最適 MPS 確率を与える情報源に対しては状態遷移（上昇）と状態遷移の（下降）との確率が等しくなるべきなので式(5-1)が成り立つ.

$$B_i \cdot p_i = q_i \qquad (5\text{-}1)$$

表 5-4　3 ビット STT-coder の LPS 幅，最適 MPS 確率，遷移 MPS 確率

		LPS 幅						最適 MPS 確率 p_i	遷移 MPS 確率 B_i
		有効領域幅							
		8	7	6	5	4	3		
確率状態	S_0	4	3	3	2	2	1	0.532	0.881
	S_1	3	3	3	2	2	1	0.574	0.742
	S_2	3	3	2	2	2	1	0.608	0.645
	S_3	3	3	2	2	1	1	0.638	0.567
	S_4	3	2	2	2	1	1	0.667	0.498
	S_5	2	2	2	2	1	1	0.698	0.432
	S_6	2	2	2	1	1	1	0.732	0.367
	S_7	2	2	1	1	1	1	0.774	0.292
	S_8	2	1	1	1	1	1	0.805	0.242
	S_9	1	1	1	1	1	1	0.909	0.100
滞在確率		0.40	0.14	0.20	0.18	0.01	0.07		

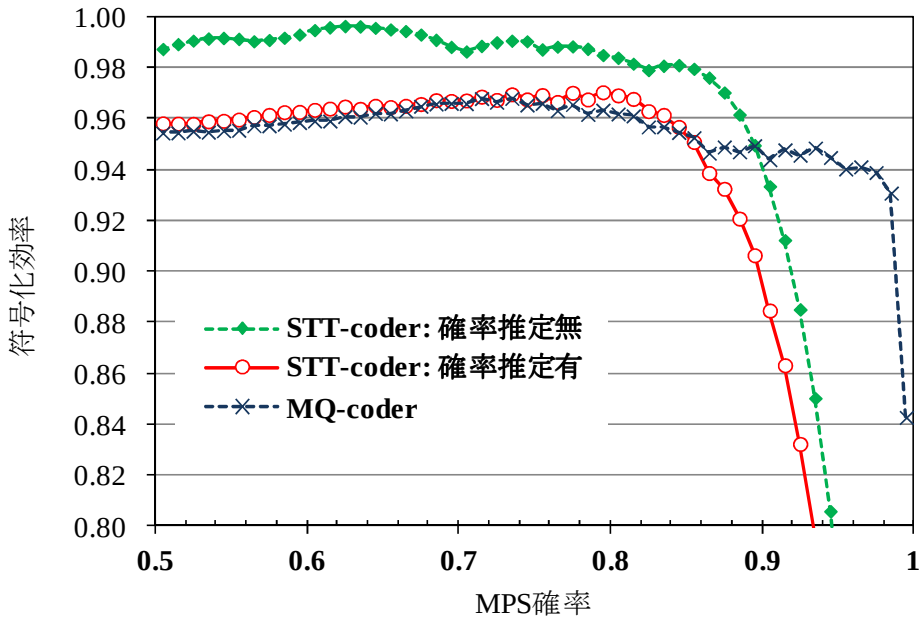


図 5-5　3 ビット STT-coder 符号化特性

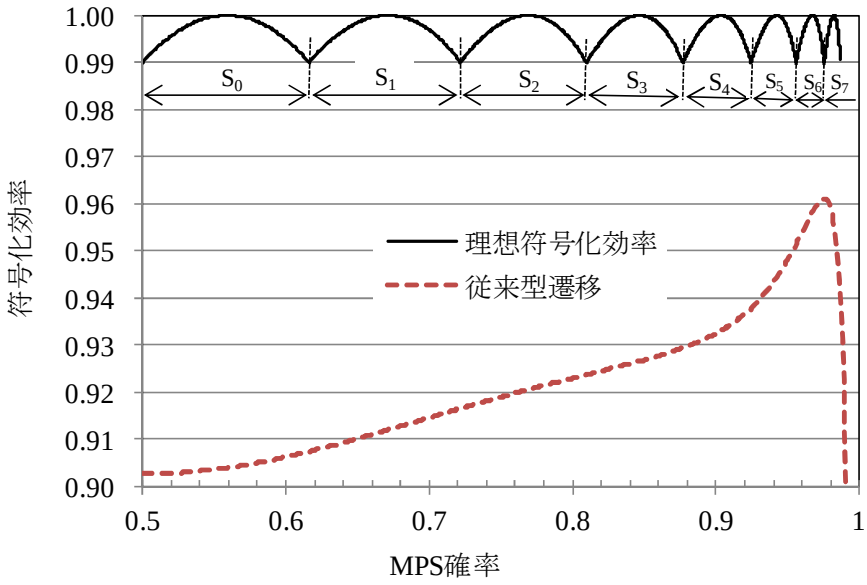


図 5-6　理想静的符号化効率

5.3.2　従来型を基本とする状態遷移型確率推定

1）3 ビット STT-coder の動的性能 [30)]

実用上最も簡易な構成の 3 ビットレジスタからなる STT-coder の設計例では，有効領域幅に対する LPS 幅の組み合わせから確率状態として表 5-4 に示す 10 状態を設定した．図 5-5 にこれら 10 状態での静的符号化性能，つまり入力される情報源に対して最適な確率状態が選択されて符号化を行った場合の符号化効率，および動的符号化性能，つまり従来の状態遷移型確率推を行った場合の符号化効率を示す．また表 5-4 には，3 ビット STT-coder の各状態における B_i，p_i を示す．この例では，各有効領域幅の発生確率を利用し，その発生確率の総和が遷移 MPS 確率になるように有効領域幅を予め選んでおき，選ばれた有効領域幅において MPS が発生した時にのみ状態遷移する処理を行っている．図 5-5 に示すように，3 ビットのレジスタ長でも MPS 確率が 0.5 から 0.9 までの情報源に対し平均で約 99%の静的符号化性能が得られることがわかる．動的性能と静的性能との効率の差は約 2.5%であり 3 ビットシステムに伴う対応範囲は狭いが，MQ-coder 等と比較しても性能的には充分高いといえる．

2）6 ビット STT-coder の動的性能 [31)]

6 ビット STT-coder では各確率状態がカバーする MPS 確率範囲は，各状態を

代表する MPS 確率 P_{Mi} に基づき，その範囲内の情報源を符号化した時に，理論的符号化効率が予め定めた設定値を超えるように定めている．図 5-6 に設定値 0.99 として 8 状態を想定した場合の符号化効率を示す. 8 状態を用意することで，MPS 確率が 0.5 から約 0.987 までの情報源に対し理想算術符号では静的な性能として最低で 99%が期待でき、4.2 で述べたように STT-coder における簡易化処理を経ても MPS 確率が 0.5 から約 0.93 までの範囲では効率低下は約 0.5%であり最低符号化効率として約 98.5%の効率が達成可能であることがわかる.

表 5-5 に 8 確率状態のそれぞれにおける B_i, p_i を示す. 次に 6 ビット STT-coder について状態遷移型確率推定を行った場合の性能測定結果を図 5-6 に従来型遷移例として示す．この図は各情報源に対し 8 確率状態の理論的滞在確率を求め，各状態での理想算術符号での符号化性能を基にシミュレーションしたものである. すなわち MPS 確率を p, 各状態での最適な情報源の MPS 確率を p_i とし，q_i＝1-p_i とすると状態 S_i における符号長 L_i は次式で与えられる.

$$L_i = p \cdot \log_2(1/p_i) + (1-p) \cdot \log_2(1/q_i) \qquad (5\text{-}2)$$

この図よりわかることは特に MPS 確率が 0.5 付近での効率が低く，MPS 確率が 0.98 付近の性能と比べると 6%程度下がっていることである．なお，ここには

表 5-5　6 ビット STT-coder における遷移 MPS/LPS 比率と有効領域幅の関係

確率状態 S_i	最適 MPS 確率 p_i	遷移 MPS 確率 B_i	B_{i+1}/B_i	理論値	実現値		理論値	実現値	
				遷移 LPS 比率 r_i	近似比率	領域幅 +ｵﾌｾｯﾄ	遷移 MPS 比率 r_iB_i	近似比率	領域幅 +ｵﾌｾｯﾄ
S_0	0.559	0.789	0.620	0.328	0.324	44	0.259	0.255	43
S_1	0.671	0.490	0.614	0.410	0.373	47	0.201	0.212	40
S_2	0.769	0.301	0.603	0.512	0.506	50	0.154	0.150	39
S_3	0.847	0.181	0.589	0.640	0.600	55	0.116	0.103	37
S_4	0.904	0.107	0.573	0.800	0.763	63	0.085	0.096	36
S_5	0.942	0.061	0.555	1.000	1.000	64	0.061	0.049	35
S_6	0.967	0.034	0.535	1.000	1.000	64	0.034	0.035	34
S_7	0.982	0.018	-	1.000	1.000	64	-	-	-

STT-coder における簡易化による効率低下は反映されておらず，動的符号化と静的符号化との差異を示したものである．

なお，STT-coder の 3 ビットシステムでは MPS 確率が 0.5 付近での動的符号化効率がそれほど低下していないのは，3 ビットシステムがカバーできる MPS 確率の範囲は 6 ビットシステムよりはるかに狭いにもかかわらず 10 状態を有しており，特に MPS 確率が 0.5 付近での状態数は 3 ビットシステムの方がきわめて密であることがその理由といえる．これと類似の傾向は 1 章でも述べたように Q-coder や MELCODE でも見られており，LPS 確率が 0.5 付近での動的性能の低下というのが従来型の状態遷移型確率推定がもつ本質的な問題であるといえる．以下ではこのような長年の問題を解決する新たな状態遷移法を提案する．

5.3.3 遷移 LPS 比率の提案

MPS 確率が 0.5 付近での動的符号化効率が低いことについては MPS が確率 0.5 付近での確率状態では，その状態に留まる確率が，MPS 確率が極めて高い確率状態における場合よりも低いことが関係していると想定できる．いま、ある確率状態に対し最適な確率をもつ情報源を考える．その情報源が最適な確率状態に滞在する確率を最大化するには，その確率状態において MPS 確率推定が 1 段階上昇する確率と 1 段階下降する確率とは同じである必要がある．したがって LPS の発生で必ず状態が下降すると想定すると、次の時点でその状態に留まる確率は $1\text{-}2q_i$（q_i: 状態に最適な LPS 確率）で与えられる．最適な状態における符号化性能と，そこからずれた状態における符号化性能の低下とが，確率状態によらずほぼ同じであるように設計されていることを前提とすると，動的性能が確率状態によらずフラットな性能を示すためには最適な状態に滞在する比率と，そこからずれた確率状態に滞在する比率とは，確率状態によらず類似の傾向を示す必要がある．しかし，従来のように LPS が発生すれば必ず状態下降が生じるように設計されていると，MPS 確率が 0.5 付近の情報源が，その最適な状態に留まる確率は MPS 確率がより高い情報源がその最適な状態に留まる確率より確実に低くなると想定される．したがって確率状態によらず最適な状態に滞在する確率を揃えることは従来の設計方針では期待できないことになる．この問題を解決するには LPS の発生で常に状態を下降させるのではなく，LPS 発生でもある比率でのみ状態を下降させるという処理が有効となる．

そこで以下では LPS についてもその発生で状態を下降させる比率を新たに導入し，その条件を満たす LPS を遷移 LPS と呼ぶ．図 5-7 に遷移 LPS を導入した

確率状態の遷移図を示す．遷移 MPS は LPS と MPS の発生比率の相違から生じているので遷移 LPS 比率の導入により新たな遷移 MPS 比率は従来の遷移 MPS 確率に遷移 LPS 比率を乗じた値が用いられる．

5.4 遷移 LPS 比率の設計

5.4.1 遷移 LPS 比率の満たすべき条件

まず，確率状態 S_i において効率が最大となる情報源を考えその情報源において，状態 S_i での滞在確率が他の状態での滞在確率より高くなるために遷移 LPS 比率が満たすべき条件を解析する．MPS 確率を p，LPS 確率を q，確率状態 S_i における遷移 LPS 比率を r_i，定常状態における滞在確率を $Z_i(p)$ とすると，定常状態では，確率状態 S_i から S_{i+1} に移る事象と確率状態 S_{i+1} から S_i に移る事象とは同回数発生する．すなわち S_i から S_{i+1} に移る確率を $P(i+1|i)$，S_{i+1} から S_i に移る確率を $P(i|i+1)$ とすると

$$Z_i(p)\,P(i+1|i) = Z_{i+1}(p)\,P(i|i+1) \tag{5-3}$$

$$P(i|i+1) = r_{i+1}\cdot q \tag{5-4}$$

$$P(i+1|i) = B_i\cdot r_i\cdot p \tag{5-5}$$

ここから，次式が成り立つ．

$$Z_{i+1}(p) = \frac{p}{q}\frac{r_i}{r_{i+1}} B_i\cdot Z_i(p) \qquad (i=0,\ldots,6) \tag{5-6}$$

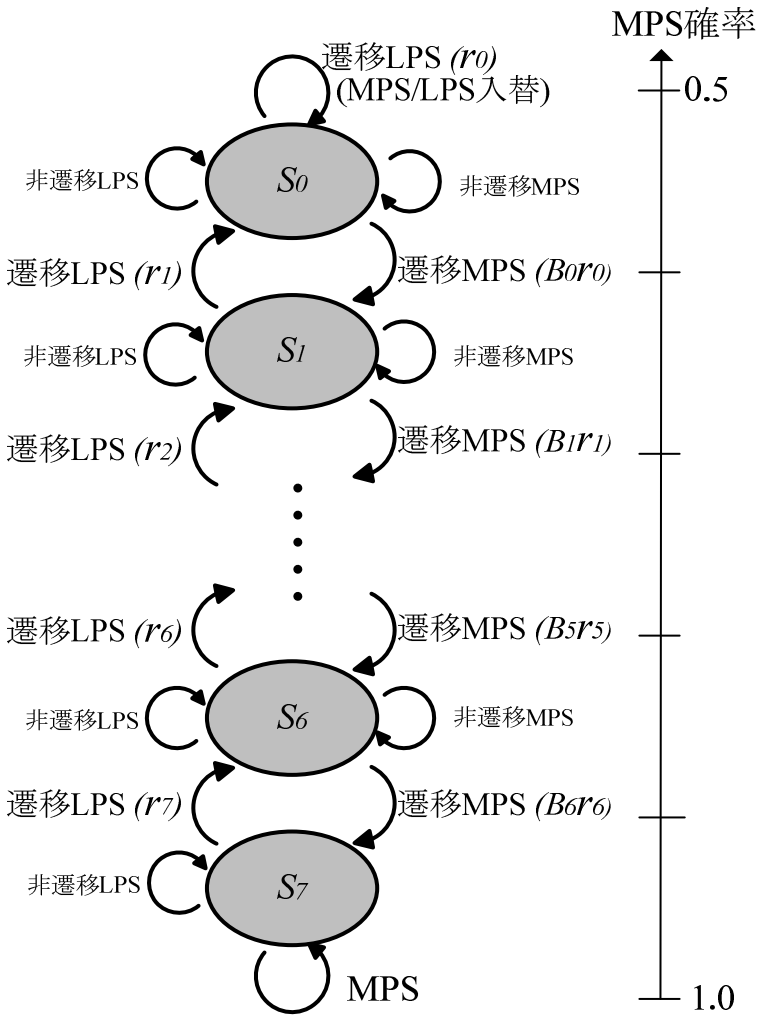


図 5-7　遷移 LPS を導入した確率状態の遷移図

このとき，確率状態 S_iにおける最適な MPS 発生確率 P_i(=1-q_i)を有する情報源が各確率状態で滞在する確率は，確率状態 S_iに滞在する確率 $Z_i(p_i)$を最大として確率状態がずれるほど減少する単峰型の分布とする必要がある．式(5-6)において，確率状態 S_{i+1}での最適確率を $p=p_{i+1}$，$q=q_{i+1}$とした場合，$p=p_{i+1}$である情報源の確率状態 S_{i+1}における滞在確率 $Z_{i+1}(p_{i+1})$が S_iにおける滞在確率 $Z_i(p_{i+1})$より大きくなるためには，次式の条件が必要となる．

$$\frac{Z_{i+1}(pi+1)}{Z_i(pi+1)} = \frac{p_{i+1}}{q_{i+1}} \frac{ri}{ri+1} B_i = \frac{B_i}{B_{i+1}} \frac{r_i}{r_{i+1}} > 1 \ (i = 0,...,6) \qquad (5\text{-}7)$$

同様に，式(5-6)において，確率状態 S_iでの最適確率を $p=p_i$，$q=q_i$とした場合，$p=p_i$である情報源が確率状態 S_iに滞在する確率 $Z_i(p_i)$を確率状態 S_{i+1}に滞在する確率 $Z_{i+1}(p_i)$より大きくする条件として次式が得られる．

$$\frac{Z_{i+1}(pi)}{Z_i(pi)} = \frac{p_i}{q_i}\frac{r_i}{r_{i+1}}B_i = \frac{r_i}{r_{i+1}} < 1 \quad (i = 0,...,6) \tag{5-8}$$

以上，式(5-7)，(5-8)より，最適な確率状態での滞在確率が最大となる条件として次式が得られる．

$$\frac{B_{i+1}}{B_i} < \frac{r_i}{r_{i+1}} < 1 \qquad (i = 0,...,6) \tag{5-9}$$

なお，B_{i+1}/B_iの値は表 5-5 に示すように，6 ビット 8 状態の STT-coder では 0.53～0.62 の値である．

5.4.2 遷移 LPS 比率の最適化

前章で解析した遷移 LPS 比率 r_i の条件をもとに，幅広い確率において符号化性能がフラットになるよう，遷移 LPS 比率を設定する．値を決定するにあたり，式(5-9)を満足するように，r_i/r_{i+1} を 0.7～1.0 の範囲で固定とし，最大値を r_7=1 として，符号化効率を検証した．情報源として MPS 発生確率が 0.5～1.0 までを 0.01 単位で想定し，各情報源を入力した際の定常状態における各確率状態に滞在する確率を計算して，それらを各確率状態での最適確率での符号化（式(5-2)による符号長）を実行して符号化効率を算出した．ここで，逓減比率 $g=r_i/r_{i+1}$ は，0.7，0.8， 0.9 の 3 種類を設定して比較した．図 5-8 に，各削減比率での符号化効率および全確率状態で遷移 LPS 比率を 1 とした場合(g=1.0)の符号化効率を示す(“8 状態”と記したカーブ)．g=0.9 は MPS 確率が高い領域で高い性能を示すが，MPS 確率が 0.5 付近での落ち込みが大きい．一方，g=0.7 では比較的広い範囲でフラットな性能が得られるが，高 MPS 確率で性能劣化が大きい．中間の g=0.8 の最悪符号化効率が最もよくフラットである．そこで，削減比率としては，g=0.8 を最適な比率と判断した．

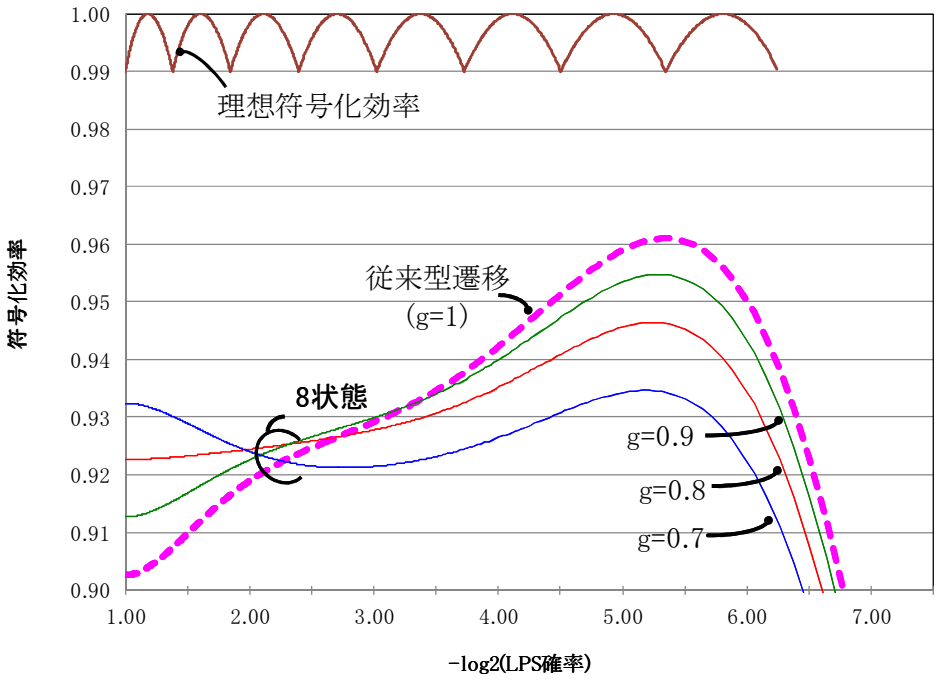


図 5-8　6 ビット STT-cder 符号化特性(1)

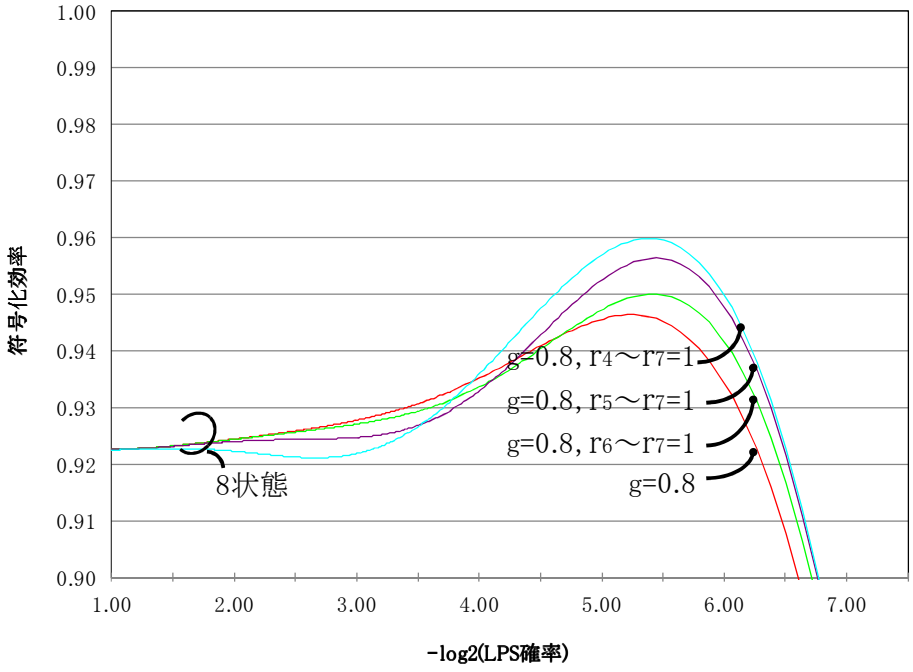


図 5-9　6 ビット STT-coder 符号化特性 (2)

しかし，この場合，高 MPS 確率においては全確率状態の遷移 LPS 比率を 1 とした場合に比べると 1%以上の効率劣化が発生している．そこで，逓減比率を固定とせず LPS 確率が高いところでは上記の 0.8 とし、高 MPS 確率では逓減比率を 1 のままとする切り替え方式を検討した．具体的には，上位 2 状態，3 状態，4 状態の遷移 LPS 比率を r_i=1 に設定する 3 パターンでの符号化効率を算出して評価した．

これら 3 パターンの遷移 LPS 比率の設定による符号化効率を図 5-9 に示す(“8 状態”と記したカーブ)．遷移 LPS 比率＝1 となる確率状態が多いほど高 MPS 確率での性能が上昇し，低 MPS 確率での性能が劣化する傾向がみられるが，最低符号化効率が維持され，平均的に高い効率を示しているのが上位 3 状態で遷移 LPS 比率＝1 とするパターンであり，これを最適な設定値と判断して採用することとする．

なお,この性能は 6 ビット 8 確率状態での理想算術符号での符号化性能であり，STT-coder では静的符号化効率の低下分として MPS 確率が 0.5 付近で約 0.5%，MPS 確率が高い部分ではマルチコンテクスト符号化に伴う,より大きな値を見込む必要がある．

5.4.3　遷移／非遷移シンボルの判定

上記図 5-8，図 5-9 の符号化効率の算出において，遷移／非遷移 MPS，遷移／非遷移 LPS の判断は，理想的な比率を想定してのものであり，実際には次のような方法により判定を行う．まず，MPS 確率が 0.5 から 0.998 までの均一分布マルチコンテクスト情報源に対し実際に符号化を行い（確率推定は理想的な推定がされると想定)，STT-coder における有効領域幅のそれぞれの発生頻度を求める．0 以外のオフセットの場合については簡易化のためオフセット値に有効領域幅を加えたものをオフセット 0 での有効領域幅に対応するものとして分布を調べる．上位 3 状態で遷移 LPS 比率を r_i=1 に設定した場合において，有効領域幅（＋オフセット）の小さい方から出現確率を累積して行った累積確率（オフセット D=0,16,24,28，および全オフセットの合計）を図 5-10，表 5-6 に示す．値によって多少の変動はあるが,領域幅の各値はほぼ均等に発生していることがわかる．ここで，全オフセットの累積確率に対して，所望の遷移 MPS（LPS）比率に最も近い有効領域幅を求める．これを各確率状態（8 通り）について記憶しておき，各確率状態に対し，有効領域幅がその値以下であったときに MPS（LPS）が出現すれば遷移 MPS（LPS）と判定し状態遷移を行う．上位 3 状態で遷移 LPS 比率

を r_i=1 に設定した場合の各確率状態での遷移 MPS 比率，遷移 LPS 比率，および上記手法により判定した場合の近似比率、近似比率を得るための領域幅＋オフセットのしきい値を表 5-5 に記載した．

以上の手法により，遷移／非遷移シンボルを判定した確率推定を実際に行ってマルチコンテクスト情報源を 6 ビット 8 確率状態の STT-coder で符号化した場合の符号化効率を図 5-11 に“実現値”と標記して示す．併せて，6 ビット 8 確率状態の理想的な算術符号を行った場合の符号化特性（図 5-9 の g=0.8, r_5～r_7=1）も“理論値”として示した．実際に符号化を行った結果では，MPS 確率が小さい範囲では理論値に近い効率を達成しているが．MPS 確率が高い範囲では効率が低下している．この理由としてマルチコンテクスト情報源を仮定したことも挙げられる。もし MPS 確率の高い情報源がシングルコンテクストで出現した場合は有効領域幅の期待値がもっと大きくなり，より高い性能が得られると思われる．ただし，MPS 確率が 1/2 に近い部分では符号化効率が落ち込んでいる．この原因として，遷移／非遷移シンボルの近似比率に理論値とズレがあることが考えられるため，特に低 MPS 確率に対応する確率状態での近似比率の影響を解析した．確率状態 S_1 において，遷移 MPS 比率の理論値が 0.201 に対し，近似比率が 0.212 と大きな差を生じている（領域幅閾値=40，表 5-5，表 5-6 参照）．そこで，遷移／非遷移の境目ではオフセットも含めて判断し，オフセット D=16 だけは，領域幅＋オフセットの閾値を一段下げて 39 に設定した．これにより，近似比率が 0.199 となり理論値との差は小さくできる．同様に，確率状態 S_1 の遷移 LPS 比率においても，オフセット D=16,24,28 において，閾値を一段上げて 48 に設定することにより，近似比率が 0.373 から 0.402 となり，理論値 0.410 により近づく．以上の判定条件の補正を加えて符号した結果を図 5-11 に“実現値（判定閾値補正）“と標記して示した．確率状態 S1 に対応する低 MPS 確率の部分で，遷移／非遷移判定をより高精度に行うことにより，符号化効率が向上することがわかる．

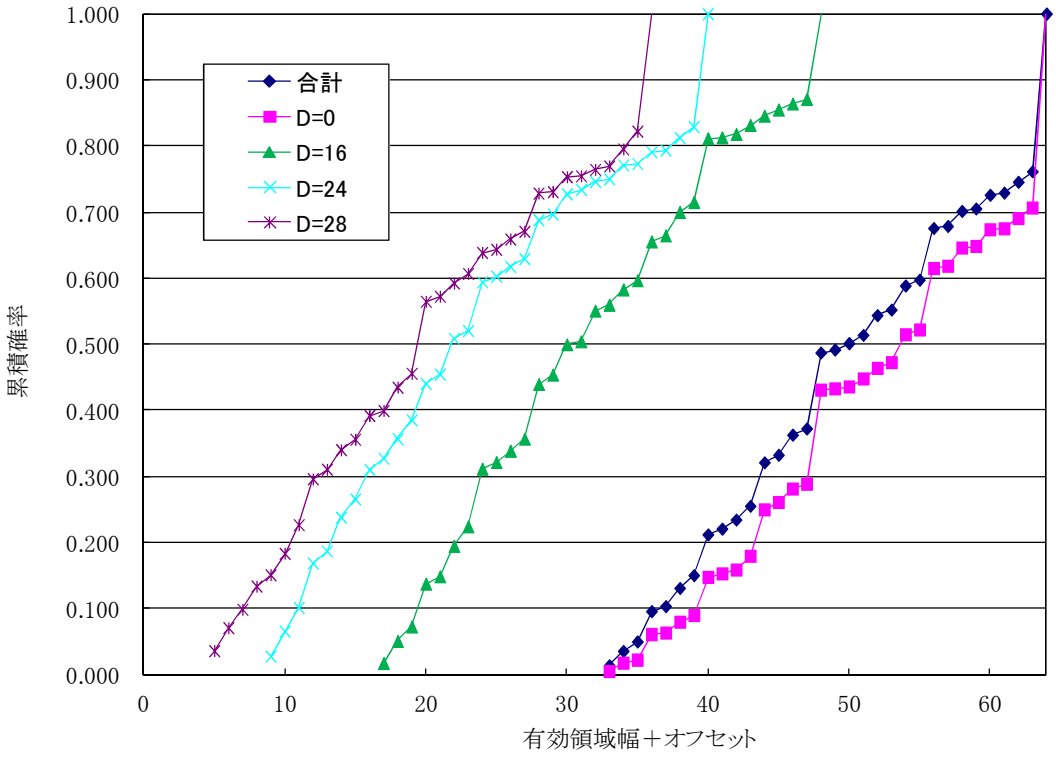


図 5-10　有効領域幅＋オフセットの頻度分布

表 5-6　有効領域幅＋オフセットの頻度分布

（累積確率は，4 種類のオフセット(D=0,16,24,28)の合計頻度に対する確率の累積値）

D=0		D=16		D=24		D=28		合計	
領域幅	累積確率	領域幅	累積確率	領域幅	累積確率	領域幅	累積確率	領域幅	累積確率
33	0.003	17	0.002	9	0.004	5	0.004	33	0.013
34	0.010	18	0.007	10	0.009	6	0.009	34	0.035
35	0.013	19	0.011	11	0.014	7	0.012	35	0.050
36	0.036	20	0.020	12	0.023	8	0.017	36	0.095
37	0.037	21	0.022	13	0.025	9	0.019	37	0.103
38	0.047	22	0.028	14	0.032	10	0.023	38	0.131
39	0.053	23	0.033	15	0.036	11	0.028	39	0.150
40	0.087	24	0.045	16	0.042	12	0.037	40	0.212
41	0.091	25	0.047	17	0.044	13	0.039	41	0.220
42	0.094	26	0.049	18	0.048	14	0.042	42	0.234
43	0.106	27	0.052	19	0.052	15	0.044	43	0.255
44	0.148	28	0.064	20	0.060	16	0.049	44	0.321
45	0.155	29	0.066	21	0.061	17	0.050	45	0.332
46	0.167	30	0.073	22	0.069	18	0.054	46	0.363
47	0.171	31	0.073	23	0.070	19	0.057	47	0.372
48	0.256	32	0.080	24	0.080	20	0.070	48	0.487
49	0.257	33	0.081	25	0.081	21	0.071	49	0.491
50	0.259	34	0.085	26	0.083	22	0.074	50	0.501
51	0.266	35	0.087	27	0.085	23	0.076	51	0.514
52	0.275	36	0.095	28	0.093	24	0.080	52	0.544
53	0.281	37	0.097	29	0.094	25	0.080	53	0.552
54	0.306	38	0.102	30	0.098	26	0.082	54	0.588
55	0.310	39	0.104	31	0.099	27	0.084	55	0.597
56	0.366	40	0.118	32	0.101	28	0.091	56	0.675
57	0.368	41	0.118	33	0.101	29	0.091	57	0.678
58	0.384	42	0.119	34	0.104	30	0.094	58	0.701
59	0.385	43	0.121	35	0.105	31	0.094	59	0.705
60	0.400	44	0.123	36	0.107	32	0.095	60	0.726
61	0.401	45	0.125	37	0.107	33	0.096	61	0.729
62	0.410	46	0.126	38	0.110	34	0.099	62	0.745
63	0.420	47	0.127	39	0.112	35	0.102	63	0.761
64	0.594	48	0.146	40	0.135	36	0.125	64	1.000

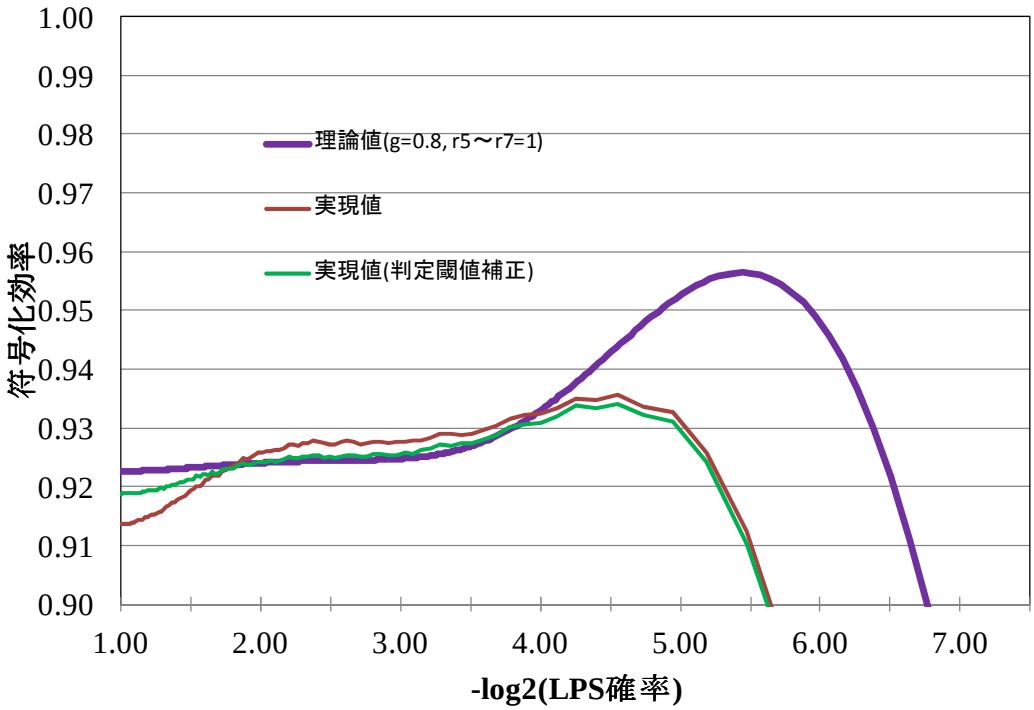


図 5-11　6 ビット STT-coder 符号化特性(3)

5.5　まとめ

本章では，最初にベイズ確率推定と状態遷移型確率推定の特性を検証し，両推定方式を適応的に切り替える状態遷移・ベイズ推定切替型確率推定を提案した．切替型確率推定は，広範囲の出現確率を有する各種情報源に対して高い符号化効率が実現できることを確認した

次に，STT-coder に確率推定を導入するにあたっては簡易化が容易な状態遷移型が適していると判断し，状態遷移型確率推定を前提とした上で STT-coder の確率推定（確率の学習）問題について考察した．これまでの課題であった LPS 確率が 0.5 付 近での性能低下の原因を究明し，符号化性能を情報源の LPS 確率に対しよりフラットにする効果を有する遷移 LPS 比率の概念を新たに導入した．確率状態 8 状態での理想符号化を前提として動的性能を解析したところ遷移 LPS 比率の導入による最低符号化効率の向上は導入前に対し約 2%であり，一定の効果を発揮することを確認できた．また，遷移／非遷移シンボルを判定するにあたり，有効領域幅の発生確率を利用する方法を導入し，実際にマルチコンテクスト情報源の符号化に適用することにより，動的性能の理論値に近い符号化効率を実現できることを明らかにした．

第6章　確率状態の 16 状態化検討

6.1　まえがき

前章で 8 状態のまま動的符号化性能を情報源の MPS 確率によらずほぼフラットにすることができたが，その性能は通常の理想算術符号を前提として 92%強であり，より高い効率の確保が望ましいといえる．前章での考察結果と従来の Q-Coder の動的効率（93%～96%）を比較すると，MPS 確率が 0.5 から約 0.982 という 6 ビットシステムでの対応情報源範囲では Q-Coder の方が状態数は 2 倍以上となっている．すなわち Q-Coder で MPS 確率が低い方から 16 番目の状態の理想 MPS 確率は約 0.93 であり、STT-coder の 8 番目の状態の理想 MPS 確率は約 0.98 である．したがって，確率推定を行う場合の状態数としては STT-coder では静的設計時の状態数を 2 倍した 16 状態としても許容されると考えられる．そこで 16 状態化を検討するに当たり，装置規模の複雑さも考慮して以下の複数の方式を検討することとする．

6.2　確率状態の 16 状態化

6.2.1　　符号化方式の 16 状態化への拡張

確率状態の 16 状態化にあたり，8 状態での各状態を 2 分割することを考える．そのため 8 状態での各確率状態の対応領域の下から 1/4，及び 3/4 の位置での値がそれぞれ最適に符号化される新たな確率状態を導入する．この 16 状態化により最低符号化効率は 8 状態での 0.99 から 0.996 に上昇する．

遷移 LPS 比率については，表 5-5 に示した 8 状態の間を新たな状態で補間して遷移 LPS 比率が一定比率で減少するよう逓減比率 r_i/r_{i+1} として $\sqrt{0.7}, \sqrt{0.8}, \sqrt{0.9}$ の場合を検証した結果，$\sqrt{0.8}$ が最適であることがわかった．また 8 状態の場合と同様に MPS 確率が高い方から数状態については逓減比率を 1 とした切り替え方式がよいということがわかり，その前提で最適化を検討した．その結果 16 状態では逓減比率を 1 とする状態数は，6 が最適となった．またこれらの確率状態がカバーする確率の範囲は，8 状態での上位 3 確率状態とほぼ同様である．図 6-1，図 6-2 にその符号化性能を示す．

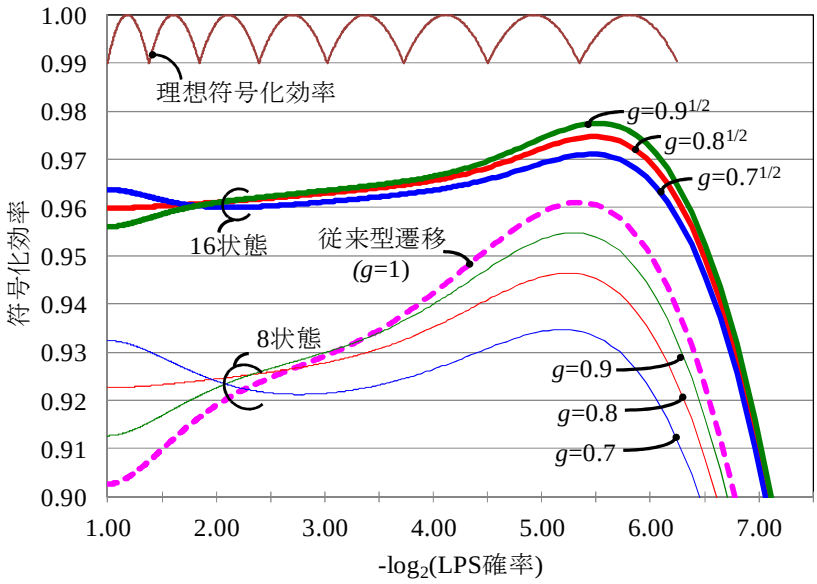


図 6-1　6 ビット STT-cder 符号化特性(16 確率状態)

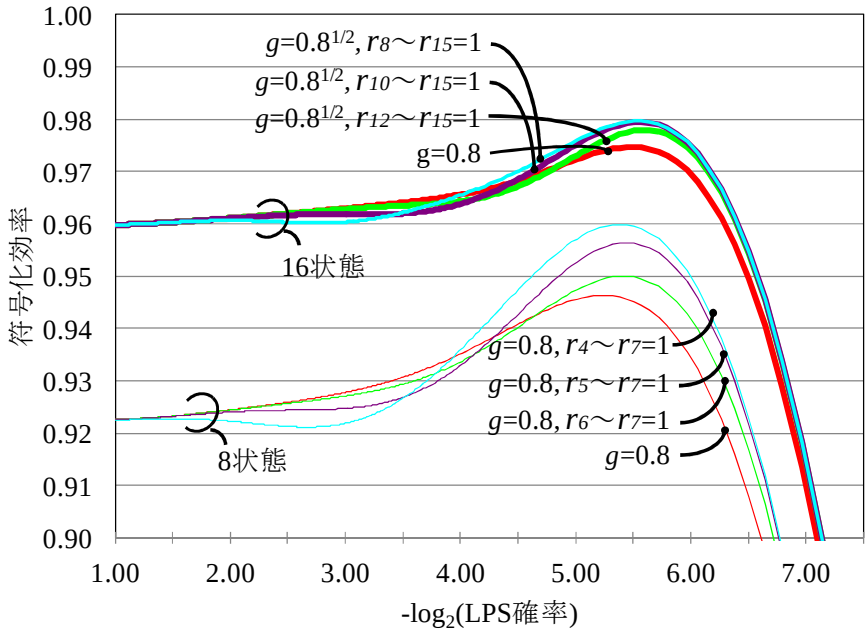


図 6-2　6 ビット STT-coder 符号化特性 (16 確率状態)

6.2.2　符号化方式の変更を伴わない簡易 16 状態化

次に符号化方式としては静的符号化性能で検討してきた 8 確率状態のままとし，8 確率状態にそれぞれ遷移の過程を示す 2 状態を設ける方式を考える．このためその連続する 2 状態では有効領域幅に対する割当 LPS 幅は共通となり，8 状態のそれぞれで最適 MPS 確率を与える情報源についていえば両状態のどちらでも最大の符号化性能を示すことになる．また状態遷移についてもその連続する 2 状態での遷移 MPS 確率と LPS 確率を対応させ，遷移 LPS 比率についてもその連続する 2 状態で同一とした．すなわち 8 確率状態方式と比較すると全く同じ性質の確率状態が 2 つ続いていることになり状態情報を 1 ビット（連続 2 状態の上か下か

表 6-1　簡易 16 確率状態の最適 MPS 確率，遷移 MPS/LPS 比率

確率状態	最適 MPS 確率 P_i	遷移 LPS 比率 r_i	遷移 MPS 比率 r_i B_i
S_{00}	0.559	0.328	0.259
S_{01}	0.559	0.328	0.259
S_{10}	0.671	0.410	0.201
S_{11}	0.671	0.410	0.201
S_{20}	0.769	0.512	0.154
S_{21}	0.769	0.512	0.154
S_{30}	0.847	0.640	0.116
S_{31}	0.847	0.640	0.116
S_{40}	0.904	0.800	0.085
S_{41}	0.904	0.800	0.085
S_{50}	0.942	1.000	0.061
S_{51}	0.942	1.000	0.061
S_{60}	0.967	1.000	0.034
S_{61}	0.967	1.000	0.034
S_{70}	0.982	1.000	0.018
S_{71}	0.982	1.000	-

の情報）余分に有しているだけということになる．これにより，各確率状態における符号化方式，つまり有効領域幅に対する LPS 幅や遷移 LPS 比率などは従来の 8 通りのままとなる．この方式を簡易 16 状態化と呼ぶこととし，表 6-1 に各状態の最適 MPS 確率，遷移 LPS 比率，遷移 MPS 比率を，また図 6-3 にはその符号化性能を求めた結果を示す．これを見ると符号化方式自体も変更した 16 状態化方式と比較して効率の低下は 0.2%にとどまっているが有効領域幅の変化や出力される符号などを規定する領域分割テーブルのサイズは通常の 16 状態化と比較して半分で済む．

したがって符号化としては 8 確率状態のままとし，動的符号化のために 1 ビット余分に状態情報をもつ簡易 16 状態符号化方式で充分であると考えられる．またアプリケーションによってはコンテクストによって静的符号化方式(非学習型)を採用したり動的符号化方式（学習型）を採用したりすることもあると思われるがその場合も両者の共通化に適しているという利点がある．

ここで領域分割テーブルのサイズを示しておくと，通常の 16 状態化では入力の確率状態（16 状態）が 4 ビット、入力シンボル（MPS/LPS）1 ビット、有効領域幅＋オフセットが（5+2）ビット、出力では領域幅＋オフセットが（5+2）ビット、出力符号語＋符号長が（6+3）ビットであるので $2^{12}\times16$ ビットの ROM を必要とする．これに対し，簡易 16 状態化では確率状態の種類が 3 ビット(8 種類)

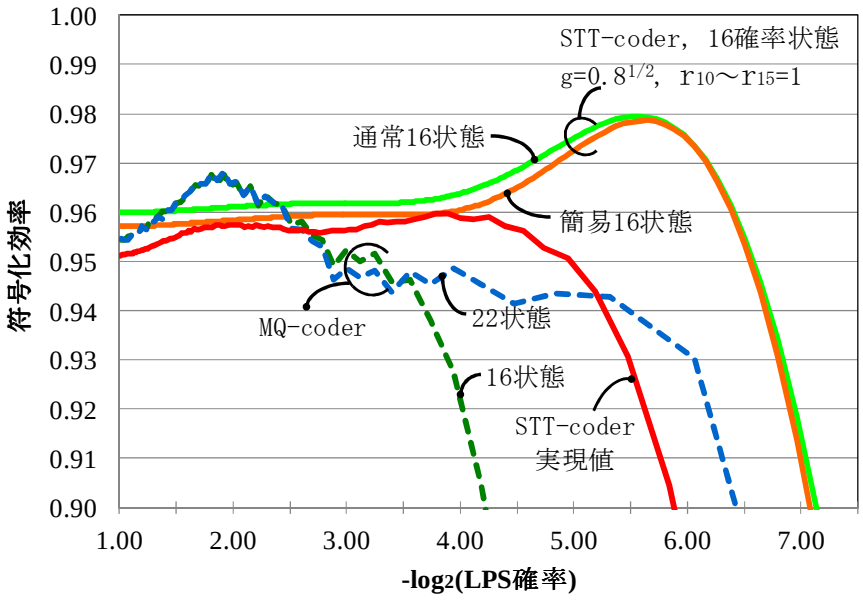


図 6-3　6 ビット STT-coder 符号化特性 (16 確率状態)

となるので，領域分割テーブルのサイズは $2^{11}\times16$ ビットですむ．なお，確率推定部に関しては現確率状態が 4 ビット、入力シンボル（MPS/LPS）1 ビット、有効領域上端アドレスが 5 ビット、出力が更新状態情報（4+1）ビットとなるので $2^{10}\times5$ ビットであるが，構成を工夫することで更にサイズを削減することも可能である．またコンテクスト数に対する確率状態記憶はコンテクスト数×5 ビットである．

簡易 16 状態方式において，マルチコンテクスト情報源に対し実際に符号化を行った結果を図 6-3 に“実現値”として示した．確率推定では，5.4.3 で述べた 8 状態での遷移／非遷移判定方法（判定閾値補正）を簡易 16 状態においても適用している．8 状態の場合と同様，MPS 確率が高い範囲を除いて，動的符号化性能の理論値に近い値を達成している．MPS 確率が 1/2 近くで僅かに劣化する傾向も 8 状態と同様である．

さて、6 ビット 16 状態の動的 STT-coder を他の算術符号と比較する上で MQ-coder で状態数を STT-coder と同一にした MQ-coder 16 状態限定版と，MQ-coder の対応する情報源の範囲を STT-coder のカバーする範囲に揃えた MQ-coder 22 状態限定版をそれぞれ作成しシミュレーションを行った．その比較結果も図 6-3 に示した．確率状態の多い MPS 確率の低い範囲では MQ-coder の性能が若干上回るが，STT-coder は簡易 16 状態化版であっても MQ-coder の 22 状態版より平均的な性能で上回ると結論できる．

6.3 まとめ

前章で述べた確率状態数 8 状態での動的符号化効率は 92%強であり，Q-coder など,より状態数の多い従来の算術符号化と比べると若干低い性能となっている．そこで STT-coder の確率状態を従来手法での状態数を超えない範囲である 16 状態に増やし，動的符号化効率の改善を図った．

16 確率状態における最適な遷移 LPS 確率を調べたところ，8 状態の場合と同様の傾向を示し，遷移 LPS の逓減比率は $\sqrt{0.8}$，逓減比率を 1 とする MPS 確率の高い状態数は 6 が最適な値となった．また，16 状態への拡張方法は，符号化方式としては 8 状態のままとして遷移の過程を示す 1 ビット情報を追加した簡易 16 状態化が通常の 16 状態化と比べ簡易でありながら十分な性能を有していることが示された．この 16 状態への拡張により，動的な符号化効率を 8 状態より約 4%改善することができた．

なお, MQ-coder で同程度の情報源範囲に対応する 22 状態版を作成し比較したところ，STT-coder の符号化効率は MQ-coder を平均的に上回り，従来方式に対する動的性能の改善が確かめられた.

第7章　結論

本論文では，ハフマン符号と同様に，状態遷移先と符号とを出力する状態遷移テーブルを参照して簡易に算術符号化処理が実行できるようにすることを目的として，算術符号 STT-coder の提案と評価を行った．本研究で得た結論を以下に要約する．

第 3 章　状態遷移テーブル参照型算術符号

- STT-coder の基礎検討として，最も簡易な 3 ビットのレジスタ長を例にとりあげ，再正規化時に必ずフラッシュを行うという簡易化案（方式 1）を考察し，フラッシュに伴う効率の低下を後続のシンボルを組み合わせて有効領域の再分割を行うことにより抑止する拡大フラッシュの導入を提案した．
- 若干のテーブルサイズの増大を許容して，最上位ビットの確定時には通常の再正規化を行う簡易化案（方式 2）を提案した．
- 上記両方式とも低 MPS 確率においては十分な符号化性能を発揮するが，高 MPS 確率においてはレジスタ長の制約に伴う性能低下が大きい．両方式の比較では，高 MPS 確率に対して性能の高い方式 2 が望ましいことを示した．

第 4 章　レジスタ長の拡張

- 高 MPS 確率の情報源に対応するため，上記簡易化案方式 2 を前提としてレジスタ長を 6 ビットに拡張し，設計パラメータであるオフセット数 N に着目して，その最適化を検討した．
- 6 ビットシステムでは，オフセット数 N=4 が符号化効率の点でほぼ最適とみなせ，その時のオフセット D の値は D=0,16,24,28 が良好な特性を与える．
- オフセットの大きな D=28,24 においては，特定の LPS 幅については現時点での符号化の最適化よりも将来の有効領域の増大化を優先させるよう領域幅の分割を変更することにより，より良好な符号化性能を実現する．
- 6 ビット STT-coder は MPS 確率が 0.5 から 0.95 の範囲の情報源に対し 98.5% 以上の符号化効率を達成する．

第 5 章　確率推定

- 簡易化に適した状態遷移型確率推定を前提とした上で，STT-coder の確率推定（確率の学習）問題について考察．これまでの課題であった LPS 確率が 0.5 付近での性能低下の原因を究明し，状態遷移における遷移 LPS 比率の概念を新たに導入した．
- 確率状態 8 状態での理想符号化を前提として動的性能を解析し，遷移 LPS 比率の導入により符号化性能を情報源のシンボル確率によらずフラットにすることができ，符号化効率の最悪値は導入前に対し約 2%向上することを示した．

第 6 章　確率状態の 16 状態化検討

- より高い動的符号化性能の実現を目指し，8 状態での各状態を 2 分割して 16 状態化するにあたり，8 状態での各確率状態の対応領域の下から 1/4，3/4 の値がそれぞれ最適に符号化される新たな確率状態を導入する方法，符号化方式としては 8 状態のままとして，遷移の過程を示す 1 ビット情報を追加して簡易的に 16 状態化する方法を検証し，簡易 16 状態化が望ましいことを示した．
- 確率状態を 16 状態化することで，動的符号化効率は 8 状態より約 4%向上すること，従来標準の MQ-coder と比較して同等以上の性能を達成することを示した．

以上，状態遷移テーブルを参照することで簡易に算術符号化を行う STT-coder について検討し，レジスタ長 6 ビットの符号器の設計アルゴリズムを確立し，その符号化性能が実用レベルにあることを示した．アプリケーションによって異なるレジスタ長の算術符号の設計が必要となることもあるであろうが，異なるレジスタ長に対しても基本的に同様の設計手法が適用可能であると推測できる．今後，より汎用的な符号器とするために，以下の課題があげられる．

- より高範囲の情報源に対する設計方法の汎用性の検証

本研究では，6 ビットのレジスタ長で方式設計を行った．必要な最大 MPS 確率がわかれば，適切なレジスタ長が導け，本検討での 6 ビットの場合と同様な方法で適切なオフセット数，オフセット値を算出し性能を評価できる．6 ビットの例からするとオフセットとして 1/4，3/8，7/16・・・という $1/2\text{-}(1/2)^n$ の系列が最

適値としての汎用性をもつかが興味深い．また，レジスタ長の具体例としてはJBIG 符号化への適用を考慮して 12 ビットレジスタが挙げられる．

・　画像符号化へ適用した場合の検証

本研究では，MPS 確率が 0.5~1.0 の情報源シンボルが一様な分布で発生することを想定している．具体的な符号化アプリケーションに適用し，適切なレジスタ長を導いたうえで，従来の算術符号と比較して，符号化性能を，検証することが望ましい．

・　確率推定における遷移／非遷移シンボル判断方法の検証

本論文では，遷移／非遷移シンボルを判断するにあたり，5.4.2 に示したように［有効領域幅＋オフセット］の値を利用した．ただし，［有効領域幅＋オフセット］の分布は，情報源の性質にある程度依存することが予想される．情報源の性質を変動させ（あるいは画像符号化に適用し），遷移／非遷移の判断基準に与える影響を解析することが必要と考えられる．

謝辞

本研究をまとめるにあたり，多くの方から多大なお力添えを頂きました．この場を借りまして，感謝の意を表します．

指導教員として研究全般で多大なご指導，ご助言を頂いた，東京工芸大学小野文孝教授に心から感謝致します．小野教授が三菱電機に在籍中から上司としてお世話になり,研究上のことのみならず公私にわたり数多くのご助言を賜りました．改めまして厚く御礼申し上げます．

また，本論文のご審査を頂き，数々の貴重なご助言を下さった，東京工芸大学木下照弘教授，浦谷則好教授，宇田川佳久教授，宇都宮大学 加藤茂夫教授に深く感謝致します．

東京理科大学伊東晋教授には，筆者が同大在学中より研究者としての姿勢，心構えから研究全般にわたり多大なご指導，ご助言を頂いているだけでなく，本研究を進めるにあったっても暖かい励ましを頂いたことに深く感謝致します．

本研究は，三菱電機株式会社情報技術総合研究所に在籍時にその機会を与えられたもので，その間多くの方々にご指導，ご鞭撻をいただきました．本研究に取り組む機会を与えていただき，貴重なご指導，ご鞭撻を賜った三菱電機株式会社山田敬喜氏，丹野興一氏，加藤嘉明氏，本研究に対して有益なご意見をいただいた吉田雅之氏，木村智広氏，高橋利至氏に深く感謝いたします．

現在の職場である三菱電機株式会社戦略事業開発室，齊藤譲室長はじめ，同開発室の皆様には，本研究を進めるにあたり暖かいご理解と励ましをいただきました．ここに厚く感謝の意を表します．

最後に社会人として博士課程に入学することを快く承諾し，常に応援してくれた妻と子供たちに心から感謝します．

2014 年 3 月

上野　幾朗

参考文献

1) D.A.Huffman: “A Method for the construction of Minimum Redundancy codes”, Proc. IRE, Vo.40, No.10, pp.1098 (1952).

2) ISO/IEC 15444-1 Information Technology -- JPEG 2000 Image Coding System: Core Coding System (2000).

3) D. Taubman, M. Marcellin: “JPEG2000 Image Compression Fundamentals, Standards and Practice”, Springer (2001).

4) D. Taubman, E. Ordentlich, M. Weinberger, G. Seroussi, I.Ueno, F. Ono: “Embedded block coding in JPEG2000”, ICIP 2000, vol.2, pp.33-36 (2000).

5) ISO/IEC 14496-10 Information Technology -- Coding of Audio-visual Objects -- Part 10: Advanced Video Coding (2010).

6) ISO/IEC 23008-2 Information Technology -- High Efficiency Coding and Media Delivery in Heterogeneous Environments -- Part 2: High Efficiency Video Coding (2013).

7) G.N.N. Martin “Range Encoding: An Algorithm for Removing Redundancy from a Digitized Message”, Video & Data Recording Conference (1979).

8) W.B. Pennebaker. J. L. Mitchell., G.G. Langdon, Jr., R.B. Arps: “An Overview of the Basic Principles of Q-Coder”, IBM J. Res. Develop., Vol.32, No.6 (1988).

9) ISO/IEC 14492 Information Technology -- Lossy/Lossless Coding of Bi-level Images (2001).

10) Elias in N.Aramson: “Information Theory and Coding”, McGrawHill, New York, pp.61-62 (1963).

11) G.G. Langdon, J. Rissanen: “Compression of Black-White Images with Arithmetic Coding”, IEEE Trans., COM-29, 6, pp.858-860 (1981).

12) ISO/IEC 11544 Information Technology -- Coded Representation of Picture and Audio Information -- Progressive Bi-level Image Compression (1993).

13) ISO/IEC 10918-1 Information Technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines (1994).

14) W. B. Pennebaker, J. L. Mitchell: “JPEG: Still Image Data Compression Standard”, Springer (1992).

15) F.Ono, S.Kino, M.Yoshida, T.Kimura:” Bi-level Image Coding with MELCODE”, Proceedings of Globecom89 , pp.255-260, Nov.1989.

16) 小野文孝，木村智行，木野茂徳，吉田雅之: “算術型 MELCODE”，信学春全体 A-152 (1989).
17) ISO/IEC 14495-2 Information technology -- Lossless and near-lossless compression of continuous-tone still images: Extensions (2003).
18) C. B. Jones: “An Efficient Coding System for Long Source Sequence”, IEEE Trans. , IT-27, 3 (1981).
19) 上野幾朗，木村智広，柳谷太一，吉田雅之，小野文孝:“シンボルカウンタ型確率推定を用いた算術型 MELCODE の高速化”， 1998 年度画像電子学会年次大会 (1998).
20) 上野幾朗，小野文孝:“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).
21) 上野幾朗，小野文孝: “再正規化時にフラッシュが可能な高速算術符号の提案”，画像電子学会研究会予稿, 11-01-08, pp.33-36 (2011).
22) 上野幾朗，小野文孝: “状態遷移テーブル参照形式による高速算術符号の検討”，画像電子学会研究会予稿, 11-02-04, pp.19-24 (2011).
23) 小野文孝: “マルコフ情報源のエントロピ符号化”，画像電子学会誌, Vol.21, No.5, pp.475-485 (1992).
24) 上野幾朗，小野文孝:“STT-coder の設計指針と静的符号化効率”, 2013 年度画像電子学会年次大会, R7-2 (2013).
25) 小野文孝，木村智広，木野茂徳，吉田雅之 :“MEL-CODE の学習型適応方式における静的特性”, 1989 年電子情報通信学会秋季全国大会,D-46 (1989).
26) I. Ueno, T. Kimura, T. Yanagiya, M.Yoshida, F. One: “Probability Estimation of Binary Information Sources for Image Coding”, ICIP 1999, vol.3, pp.797-801 (1999).
27) 上野幾朗，木村智広，小野文孝:“状態遷移・ベイズ推定切替型確率推定とその画像符号化への適用”，電子情報通信学会論文誌 A，Vol.J83-A，No.6, pp.722-725 (2000).
28) 木村智広，上野幾朗，柳谷太一，吉田雅之，小野文孝:“2 値情報源の適応エントロピ符号化における出現確率推定方式 “, 信学技報, IE96-135 (1997).
29) ISO/IEC 14495-2 Information Technology - Lossless and Near-lossless Compression of Continuous-tone Still Images – Extension (1999).
30) 上野幾朗，小野文孝:“状態遷移テーブル参照型高速算術符号の動特性”, 2012

年度画像電子学会年次大会, R6-4 (2012).
31) 上野幾朗, 小野文孝:“状態遷移テーブル参照型算術符号 STT-coder における確率推定方式と動的符号化の設計”, 画像電子学会誌, Vol. 43, No. 1, pp.71-78 (2014).

発表文献一覧

学会誌論文（査読有）

(1) 上野幾朗，木村智広，小野文孝，“状態遷移・ベイズ推定切替型確率推定とその画像符号化への適用”，　信学論 A, Vol.J83-A, No.6, pp.722-725 (2000).

(2) 上野幾朗，小野文孝: “状態遷移テーブル参照型算術符号（STT-coder）の設計”，画電学会誌 Vol.43, No.1, pp.62-70 (2014).

(3) 上野幾朗，小野文孝: “状態遷移テーブル参照型算術符号(STT-coder)における確率推定方式と動的符号化性能”，画電学会誌 Vol.43, No.1, pp.71-78 (2014).

国際会議（査読有）

(1) Ikuro Ueno, Fumitaka Ono: “An adaptive binary arithmetic coder using a state transition table”, IEVC 2012, 5C-1 (Nov. 2012).

(2) Ikuro Ueno, Fumitaka Ono: “Fast Arithmetic Coder Driven by State Transition ROM Table”, IWAIT 2013, 2C-3 (Jan. 2013).

(3) Ikuro Ueno, Fumitaka Ono: “Design Principle and Static Coding Efficiency of STT-coder: a Table-Driven Arithmetic Coder”, ICSPS 2013, C-023 (Dec. 2013).

学会発表等

(1) 上野幾朗, 小野文孝: “再正規化時にフラッシュが可能な高速算術符号の提案”, 画電学会研究会, 11-01-08, pp.33-36 (Aug. 2011).

(2) 上野幾朗, 小野文孝: “状態遷移テーブル参照形式による高速算術符号の検討”, 画電学会研究会, 11-02-04, pp.19-24 (Oct. 2011).

(3) 上野幾朗，小野文孝: “状態遷移テーブル参照型高速算術符号の動特性”,第 40 回画電年大, R6-4, (Jun. 2012).

(4) 上野幾朗，小野文孝: “STT-coder の設計指針と静的符号化効率”,第 41 回画電年大, R7-2 (Jun. 2013).