The 27th Symposium on Information Theory
and Its Applications (SITA2004)
Gero, Gifu, Japan, Dec. 14-17, 2004

辞書番号を修正したLZW符号
LZW Algorithm Modifying the Reference Number on the Dictionary

吉田 幸二* 石田 崇* 平澤 茂一*
Koji YOSHIDA Takashi ISHIDA Shigeichi HIRASAWA

Abstract— In the field of data compression, it is known that the LZW algorithm achieves the asymptotically optimum compression even when the statistical property of input sequence is unknown. Here, we consider a symbol added in the case a dictionary is registered. Its symbol is the first symbol of a partial data sequence given by next incremental parsing. That is, this symbol can specify the dictionary number referred to next. In this paper, we propose a new algorithm modifying the dictionary number so that an independent dictionary number may be assigned for every symbol registered at first, and show by simulation results that it attains lower compression rate than that of the conventional method .

Keywords— data compression, LZW algorithm, dictinary, reference number

1 はじめに

近年，マルチメディアの発展に伴い，我々が扱うデータは多様化し，その量も増加の一途を辿っている．そこで，多種多様なデータを効率よく記憶，伝送するための技術として，データ圧縮の重要性が高まっている．データを生成する情報源の確率構造が既知の場合，算術符号化，ハフマン符号化等により漸近的にエントロピーまでの圧縮が可能である [1]．しかし現実的な問題として，情報源の確率構造は未知である事が多く，入力系列の統計的性質が未知でも入力系列が長くなるにつれて漸近的に最良の圧縮が可能なユニバーサル符号が注目されている．代表的なユニバーサル符号化法の一つであるLZ78法 [2] を改良したLZW法 [3] は，あらかじめ1記号からなる語をすべて辞書に登録しておくことで，常に登録されている記号列の辞書番号のみを符号化する手法である．新たな記号列を辞書に登録する際，現在の最長一致系列に次の1記号を加えた記号列を辞書に登録する．ここで，加えられた1記号は，次に増分分解される記号列の最初の1記号となる．これより，増分分解して得られる記号列に割り当てられている辞書番号は，最初の1記号を辿った先の辞書番号に限定される．すなわち，当該1記号先に登録されている辞書の数だけ番号を用意すれば足りるということがわかる．

そこで本研究では，辞書に登録する際の次の1記号を考慮し，増分分解して得られる辞書番号を修正する方法を提案する．また，提案符号化法のベンチマークデータ (Calgary corpus [7]) への適用によって，従来のLZW符号化法に比べ良い圧縮率を達成することを示す．

2 従来手法

本節ではLZW法のアルゴリズムを示す．

*〒 169-8555 東京都新宿区大久保 3-4-1 早稲田大学理工学部経営システム工学科 School of Science and Engineering, Waseda University Okubo 3-4-1, Shinjuku-ku, Tokyo, 169-8555 Japan. E-mail: yoshida@hirasa.mgmt.waseda.ac.jp

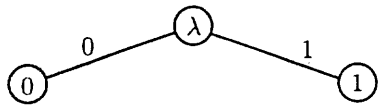

図 1: M=2 のLZW法による辞書の初期設定

2.1 LZW法 [3]

現在までに開発された圧縮アルゴリズムの中で，理論的にも実用的にも最も重要な圧縮法の一つであるLZ78法には多数のバリエーションが存在する．その中でも，最も基本的で広く用いられている圧縮法が，1984年にDEC社のT.Welchによって開発されたLZW法である．

増分分解に基づくLZ78法では，第jステップの符号には，末尾の文字$x_{n_j} \in \{0,1,\ldots,J-1\}$を含んでいた．そして，この末尾の文字は何の圧縮操作も施されずにそのまま復号器に渡されるだけであったが，Welchはこれによる効率の低下を防ぐ方法を提案した．それはあらかじめ1記号からなる語をすべて辞書に登録しておくことで，符号語中に含まれる記号を取り除き，常に登録されている記号列の辞書番号のみを符号化する手法である．

なお，LZW法でもLZ78法と同様に，入力系列を増分分解という方法で部分記号列に分解する．得られた部分記号列をトライと呼ばれる文脈木による辞書に登録し，この辞書を利用して過去の部分記号列をコピーすることによって圧縮を行う．増分分解アルゴリズムによって得られた記号列の中間符号語を，部分記号列の数だけに依存する桁数の2進数で表す．そして全体の入力系列に対する符号語は，これらの2進数を連接したものとする．

また，LZW符号においても，定常情報源クラスに対して，データ系列が無限長のとき平均符号長がエントロピーに収束（漸近最良性）するユニバーサル符号であることが示されている [4]．

2.2 LZW法のアルゴリズム

情報源アルファベットを$A=\{a_1,a_2,\ldots,a_M\}$とする．現在符号化を行っている記号の記号列内での位置を示すポインタをiとし，既に辞書に登録されている記号列の数，すなわち節点の数を表す変数をjとする．

以下にLZW法の符号化アルゴリズムを示す [3]．

(step1) 初期設定

辞書を表す木として，根とM個の節点（節点番号0～$M-1$）をもつ木を用意する．根と番号$k-1$ ($1 \le k \le M$)の節点を結ぶ枝にはラベルa_kを付ける．また，iを$i \leftarrow 1$に設定し，jを$j \leftarrow M$とする．

(step2) i番目の符号語の構成

i番目から始まる記号列に対し，最長一致系列を辞書中から探す．実際には，記号列の各記号に従って，根から順に枝のラベルをたどり，たどれなくなったところが最長一致系列である．最長一致系列の辞書番号r_j ($0 \le$

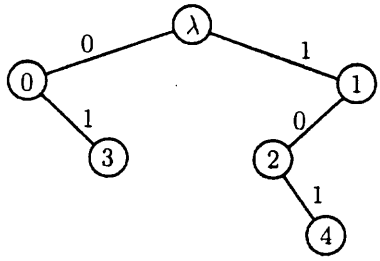

図 2: 入力列=101011（M=2）のLZW法による符号語の構成

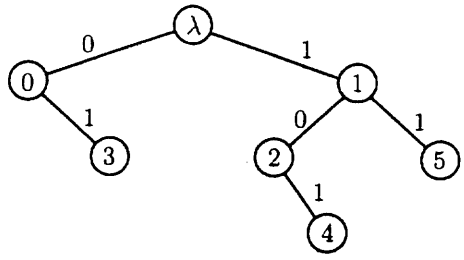

図 3: 入力列=101011（M=2）のLZW法による辞書の更新

$r_j \leq j-1$）を算術符号で表したものを符号語として出力する．次に，iを$i \leftarrow i+l$に更新する．ただし，lは最長一致系列長である．

図2の例は，記号列1010に対応する符号語の構成を終えた辞書の状態である．このとき，$i \leftarrow 5$，$j \leftarrow 5$となっている．ここで記号列11を読み込むと，節点1までたどることができるので，辞書番号1を算術符号で表したものを符号語として出力する．次に，iを$i \leftarrow 6$に更新する．

(step3) 辞書の更新

新たな節点をつくり，番号jを与える．また，この節点jと節点r_jとを新たに枝で結び，枝にラベル$x_j \in A$（次に符号化される1記号）を付ける．次に，jを$j \leftarrow j+1$に更新した後，(step2) へ戻る．

図3の例では，新たな節点をつくり番号5を与えている．このとき，$i \leftarrow 6$，$j \leftarrow 5$となっている．この節点5と節点1とを新たに枝で結び，枝にラベル1を付ける．次に，jを$j \leftarrow 6$に更新する． □

2.3 適応型算術符号 [5]

適応型算術符号は，記号列を実数0と1の間の実数の区間を用いて表す．この区間が各記号の出現頻度によって定まる生起確率に比例した小区間に分割される．記号を読み込むごとに各記号の生起確率（または出現頻度）を変更し，現在読み込んだ記号列に対応する区間を算出しながら符号化を行う．最終的に得られた区間に含まれる一つの数を2進数表現したものが符号語となる．

以下，アルゴリズムを示す [5]．

① 情報源アルファベットを既知とし，情報源アルファベット中の各記号の生起確率が等確率になるよう初期値を与える．

② 記号列を読み込み，読み込んだ記号に対応する区間を算出し，読み込んだ記号の出現頻度を更新する．

③ 各記号の生起確率と記号列に対応する小区間を再計算し②へ．なお，次の符号化では更新した区間を用いる． □

3 提案手法

3.1 提案手法への着眼点

LZW法では辞書更新の際，最長一致系列に次の1記号を加えた記号列を辞書に登録する．ここで加えられる1記号は次に増分分解される記号列の最初の1記号であることに着目する．最長一致系列を表す辞書番号は，増分分解される最初の1記号を辿った先にある辞書番号に限定され，その他の記号を辿った先にある辞書番号が参照されることはない．これより最長一致系列の番号を参照する際，辞書登録時に加えた1記号を辿った先にある辞書のみに番号が割り振られていればよいということになる．すなわち，初期登録されている1記号ごとに独立な辞書番号を割り振るよう辞書番号を修正することができる．しかし，異なる記号列に同じ辞書番号が存在するので，どの記号列に対する辞書番号なのかを特定するために最初の1記号を出力する必要がある．

3.2 提案手法の概要

提案手法においてもLZW法と同様に，符号化の開始時において，符号器と復号器はアルファベットのすべての文字M個からなる辞書を用意する．提案手法では従来のLZW法とは異なり，増分分解された最長一致系列を表す中間符号語として，増分分解された最初の1記号とその記号の先に登録されている辞書のうち最長一致した辞書の番号を出力する．最終的には最初の1記号と参照された辞書番号を算術符号化して符号語とする．また，辞書中の記号列には，その記号列の辞書番号とその記号列の辞書番号が符号化された回数（参照回数）を記憶し，符号語を出力してから辞書を更新するまでの間に符号化された記号列の参照回数に1を加え，これを算術符号化する際の出現頻度に用いる．

辞書を更新する際，新たな記号列に割り振る辞書番号は，現在増分分解されている記号列の最初の1記号先に登録されている最大の辞書番号に1を加えた番号である．

3.3 提案手法のアルゴリズム

2.2節のLZW法のアルゴリズムと同様に，情報源アルファベットを$A=\{a_0, a_1, \ldots, a_{M-1}\}$とする．現在符号化を行っている記号の記号列内での位置を示すポインタをiとし，増分分解される最初の1記号をa_kとする．また，最初の各記号の先にある辞書番号のうち，最大の辞書番号をそれぞれu_k（$0 \leq k \leq M-1$）とする．次の辞書の更新における新たな節点の辞書番号はu_k+1となる．最長一致系列の節点番号をr_k（$0 \leq r_k \leq u_k$）とする．

提案手法では，符号化後に最長一致系列の節点の参照回数に1が加わるが，1を加える前の節点r_kの参照回数をt_{r_k}，1を加えた後の節点r_kの参照回数をT_{r_k}とする（$T_{r_k} = t_{r_k} + 1$）．

提案手法の符号化アルゴリズム

(step1) 初期設定

辞書を表す木として，根と M 個の節点（辞書番号 0）をもつ木を用意する．また，すべての節点に辞書番号 0 と参照回数 0 を与える．根と番号 0 の節点を結ぶ枝にはラベル a_k を付ける．辞書内の最初の 1 記号以下の最大の辞書番号 u_k $(0 \leq k \leq M-1)$ を 0 にする．さらに，i を $i \leftarrow 1$ に設定する．

(step2) i 番目の符号語の構成

増分分解される最初の 1 記号 a_k と最長一致系列の辞書番号 r_k とを算術符号で表したものを符号語として出力し，i を $i \leftarrow i+l$ に更新する．

(step3) 辞書の更新

新たな節点をつくり，番号 u_k+1 を与える．また，この節点 u_k+1 と節点 r_k とを新たに枝で結び，枝にラベル $x_k \in A$（次に符号化される 1 記号）を付ける．次に，u_k を $u_k \leftarrow u_k+1$ に更新した後，(step2) へ戻る． □

(step2) において，原則として出力される中間符号語は増分分解される最初の 1 記号と最長一致系列を表す辞書番号であるが，増分分解される最初の 1 記号を出力する必要がない場合がある．以下，この例外について説明する．

例外) 増分分解される最初の 1 記号を出力する必要がない場合

提案手法では増分分解される最初の 1 記号ごとに辞書番号が割り振られる．この時，辞書内には異なる記号列に同じ辞書番号が割り振られている．そこで，どの記号列に対する辞書番号かを特定するために最初の 1 記号を出力する必要がある．しかし，最初の各記号ごとの最大の辞書番号 u_k の中から 2 番目に大きい辞書番号を記憶しておき，これより大きい辞書番号が参照された場合には，その番号は辞書内で一つしかない番号なので最初の 1 記号を出力する必要がない．

3.4 提案手法のアルゴリズムの例

本節では，従来手法のアルゴリズムの例と同様に情報源アルファベット A={0,1} (M=2) において，入力系列 1010101 を提案手法により符号化する例を示す．

(step1) 初期設定

図 4 は提案手法の辞書の初期設定の様子を示す．四角形枠の左は辞書番号，右は参照回数を表す．

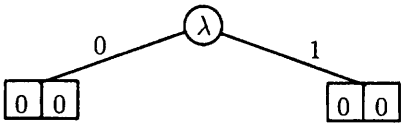

図 4: M=2 の提案手法による辞書の初期設定

(step2) i 番目の符号語の構成

図 5 の例は，記号列 101 を読み込んだ後の辞書である．このとき，$i \leftarrow 3$ となっている．ここで記号列 10 を読み込むと，節点 1 まで辿ることができ，さらに次の記号 0 を読み込むと番号 1 の節点からはラベル 0 の枝がないので辞書番号 1 と増分分解された最初の 1 記号 1 を中間符号語として出力し，i を $i \leftarrow 4$ に更新する．そして辞書番号 1 の参照回数に 1 加え 1 とする（図 6 参照）．[1]

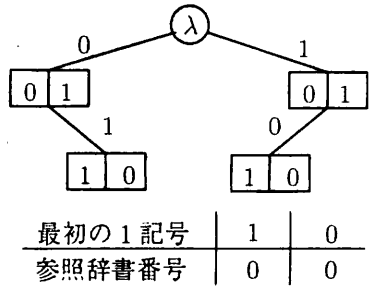

図 5: 入力列=101 (M=2) の提案手法の辞書

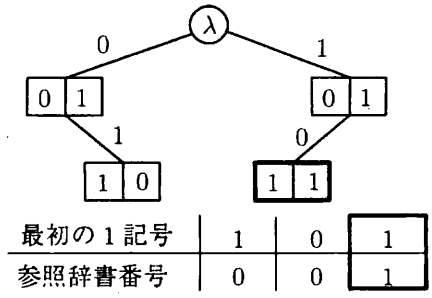

図 6: 入力列=1010 (M=2) の提案手法の辞書

図 7 では，中間符号語を出力する際の例外処理の例を説明する．今，記号列 10101 まで入力した後の辞書がある．次に記号列 101 が入力されると増分分解される最初の 1 記号である 1 と最長一致系列の辞書番号である 2 が出力されるが，この辞書番号 2 は各記号ごとの最大の辞書番号 u_k の中で 2 番目に大きい辞書番号 (u_0= 1) よりも大きい数である．よって，最初の 1 記号である 1 を出力しなくてもどの記号列に対応する辞書番号かを特定することができるので，最初の 1 記号である 1 を出力する必要がない．

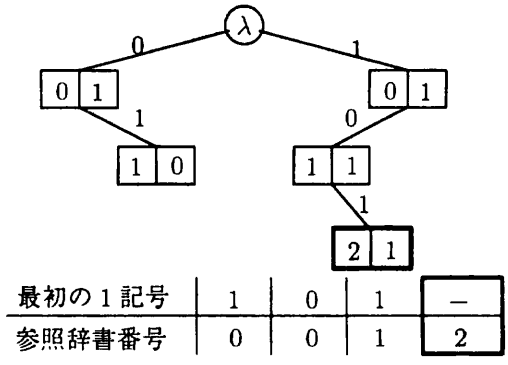

図 7: 入力列=1010101 (M=2) の提案手法の辞書

(step3) 辞書の更新

図 8 の例は，図 6 の状態から記号列 101 を新たに辞書登録する様子を表している．このとき，$i \leftarrow 4$，$u_1 \leftarrow 1$ となっている．なお，辞書の更新後 $u_1 \leftarrow 2$ となる．

[1] 図 6 において現在参照された節点と現在出力された中間符号語は太字で縁取りされている．

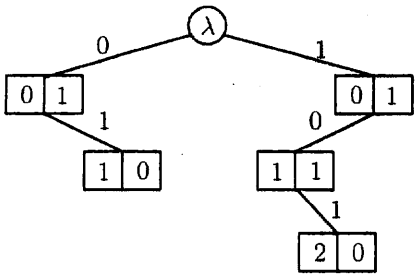

図 8: 入力列=1010101（M=2）の提案手法による辞書の更新

4 シミュレーション結果

本節では，提案手法の有効性を明らかにするため，以下の条件によりシミュレーションを行う．

Calgary corpus [7] に対して，LZW 法，及び提案手法を適用し圧縮率を求める．それぞれの圧縮結果を表 1 に示す．なお，中間符号語から符号語への変換については，両手法ともに適応型算術符号を用いた．情報源アルファベット数 $M = 256$ に設定した．

表 1: シミュレーション結果 [圧縮率]

データ	byte	LZW 法	提案手法
bib	117543	0.3818	0.3803
book1	785394	0.3879	0.3864
book2	268843	0.3892	0.3897
geo	102400	0.6361	0.6256
news	387170	0.4365	0.4337
obj1	21504	0.5758	0.5646
obj2	246814	0.4885	0.4810
paper1	54413	0.4425	0.4395
paper2	83932	0.4127	0.4113
paper3	47628	0.4464	0.4444
paper4	13582	0.4813	0.4794
paper5	12276	0.5041	0.4989
paper6	39126	0.4541	0.4505
pic	513216	0.1137	0.1101
progc	41100	0.4423	0.4347
progl	71908	0.3592	0.3536
progp	51347	0.3606	0.3547

5 考察

5.1 圧縮率

(1) 圧縮率

表 1 より，提案手法ではすべてのファイルにおいて圧縮率の改善が見られた．これは，辞書内の最大の辞書番号が小さくなったために各辞書番号の出現頻度が上がり，算術符号化を行うのに適していると考えられるからである．また，増分分解される最初の 1 記号においても出力しなくてよい記号を考慮し，冗長性を取り除くことができたので圧縮率を改善できたとも考えられる．

(2) 辞書番号と出力しない最初の 1 記号

表 2 では，提案手法の圧縮率を最も改善できたファイル obj1 と，提案手法の圧縮率を最も改善できなかったファイル paper2 について最大の辞書番号の従来との比較と最初の 1 記号を出力しなかった回数を表している．obj1 では，提案の最大の辞書番号が従来の最大の辞書番号より約 6 %にまで小さくすることができている．これより，出力される辞書番号の範囲が小さくなり，算術符号に好影響を与えたと考えられる．また，最初の 1 記号を出力しなくてよい回数が全出力回数の 3 %を超え，他のファイルよりこの割合が大きいことからも圧縮率が改善された理由がわかる．paper2 では最初の 1 記号を出力しなくてよい回数が全出力回数の約 1 %しかない．また，提案の最大の辞書番号は従来の最大の辞書番号と比べて約 9 %の小ささになっているが，他のファイルと比べるとこの値は大きい．これらの理由より，圧縮率がそれほど改善できなかったのではないかと考えられる．

表 2: obj1 と paper2 の辞書番号と出力しなかった回数の比較

	obj1	paper2
従来の最大辞書番号	9324	21734
提案の最大辞書番号	568	1972
出力しなくてよい回数	301	345

5.2 計算量

提案手法では，従来手法と比べて出力する中間符号語が一つ増えているので計算量が増加する．具体的には，中間符号語が一つ増加することにより算術符号化の確率計算が 2 倍になるだけである．その結果，漸近的な計算量としては従来手法と同じオーダで提案手法を実現できたと言える．

6 まとめと今後の課題

本稿では，辞書登録する際の次の 1 記号に着目し，増分分解される最初の 1 記号ごとに辞書番号を修正する手法を提案した．これより辞書の最大辞書番号を小さくすることを可能にし，Calgary corpus [7] に対してシミュレーションを行うことにより提案手法の有効性を示した．一方，辞書の最大辞書番号を小さくすることはできたが，増分分解される最初の 1 記号を出力する必要があるため，予想していたほどの圧縮率の改善はできなかった．

今後の課題としては，増分分解される最初の 1 記号の出力を考慮して圧縮率の改善を図るアルゴリズムの提案をすることである．

7 謝辞

著者の一人である吉田は，本研究を行うにあたり，数多くのご助言，ご支援を賜りました早稲田大学平澤研究室の各氏に感謝いたします．

参考文献

[1] 韓　太舜，小林欣吾，情報と符号化の数理，培風館，1999.
[2] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding", *IEEE Trans. Infor. Theory* , Vol.IT-24, No.5, pp.530-536, 1978.
[3] T.A.Welch, "A Technique for High-Performance Data Compression", *IEEE Computer* , Vol.17, No.6, pp.8-19, 1984.
[4] H. Morita, "Asymptotical Analysis of LZW Coding Algorithm", *Proc of the 12th Symposium on Information Theory and Its Applications*, pp.541-546, 1989.
[5] 植松 友彦，文書データ圧縮アルゴリズム入門，CQ 出版，1995.
[6] 情報理論とその応用学会，情報源符号化＝無歪みデータ圧縮，培風館，1998.
[7] ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus