辞書配列を利用した非モード方式によるシフトJIS文書圧縮

A Non-Modal Type of Shift-JIS Text Compression by Using A Dictionary Array

伊藤 雅 †
Masaru ITOH

Abstract This paper proposes a new data compression method for a Japanese-text file, where the text is written in shift-JIS (JIS X 0208) codes. In the first pass, a dictionary array is built up by the higher frequency both single and multi-byte characters. Then in the second pass, the dictionary items substitute all the registered characters. The code 0xFF is put into a compressed file in front of non-registered character so as to distinguish non-registered characters from registered ones. It takes $O(1)$ time on a hashing basis to confirm whether each input character is in the dictionary, and to transfer its code to a dictionary item. Furthermore, the run-length coding apply to a sequence of successive spaces for the purpose of accomplishment of the much higher compression ratio. The code 0xFE is used in this coding. A feature of the method is to be a non-modal type of compression.

1. はじめに

国内外で文書データ圧縮に関して盛んに研究されている[1,2]．しかし対象を日本語文書に限った研究は少ない．荻原[3]はJISコード体系で仮名文字のみを対象とする短縮表現を提案した．これはモード式符号化方式であるJISコードに連続仮名モードを追加して圧縮を実現する方法である．伊藤ら[4]は非モード式符号化方式のシフトJISコードに2バイト系モードを導入して荻原の短縮表現をシフトJISコード体系に拡張した．モード切り替えにはシフトJISの未定義領域のコードをシフトイン・シフトアウトとして利用する．更に文献[4]では，漢字を含む高出現頻度の2バイト文字を辞書配列に登録し，辞書項目で2バイト文字を置き換えて短縮表現する圧縮も提案している．

シフトJISコードに2バイト系モードを導入した場合の欠点のひとつは随所にシフトイン・シフトアウトが挿入される点である．文書によってはファイルサイズの数%が占められることもある．もうひとつの欠点は元文書の文字列情報がシフトイン・シフトアウトで分断され破壊される点である．

以上の欠点を克服するために，非モード式符号化方式のままでシフトJIS文書を短縮表現圧縮する方法を提案する．提案法はテキストベースの表組みで多用される連続スペースにも対応している．本短縮表現圧縮は単独使用でも有効である．この短縮表現化を既存圧縮法の前処理に利用すれば，既存圧縮法を単独で使用する場合の圧縮率を更に改善することもできる．

†愛知工業大学 経営工学科 （豊田市）

2. シフトJISコードと対象文字セット

シフトJISコードは国内のパソコンをはじめ各種プラットフォームの内部コードとして広く採用されている．JIS X 0208[5]の付随書1で「シフト符号化表現」として規定されている．JIS X 0208の文字セットには漢字6355字，非漢字524字の計6879字が標準文字として収められている．漢字は第一水準漢字2965字と第二水準漢字3390字から構成される．提案法の対象文字セットはこの6879字に限定する．よってJIS X 0212[6]で規定される6067字の文字セットは対象外となる．シフトJISコードは半角片仮名とASCII/JISローマ字もサポートしている．シフトJISコードの仕様を表1に示しておく．特筆すべきはJIS X 0208がすべての区点上で文字を規定している訳ではない点である．例えば第85区以降，シフトJISでは0xEB以降が現在，未定義となっている．この領域内のコード0xFEを連続スペース圧縮に，コード0xFFを仮想2バイト文字化に利用する．情報交換用ではなく圧縮用として利用するので規格にも抵触しない．

日本語符号化方式にはシフトJISコード以外にもJISコード，EUC（Extended UNIX Code），Unicodeなどが存在する．JISコードは電子メールや電子ニュースなど情報交換用コードとして特に優れている．理由はエスケープシーケンスによるモード式符号化方式だからである．EUCはマルチバイトコードをサポートしており，UNIXマシンに実装されることが多い．UnicodeはJIS X 0221[7]の中ではUCS（Universal Multiple-Octet Coded Character Set）として規定されている．

表1 シフトJISコードの仕様

用　途	コード値
2バイト文字の第1バイト範囲	81~9F, E0~EF
2バイト文字の第2バイト範囲	40~7E, 80~FC
半角片仮名	A1~DF
ASCII/JISローマ字	21~7E

これは多国語言語からなるすべての文字を一律1文字16/32ビット（UCS-2/UCS-4）で表現する符号化方式である．JIS，EUC，シフトJISの3符号化方式についてはJISコードを介して相互に変換可能である．変換アルゴリズムの詳細については文献[8]を参照されたい．Unicodeの仕様は画期的ではあるが他の3コードへの変換が容易ではなく，また，フォントの問題も存在し，普及には至っていない．

日本語文書に限定すれば，これら符号化方式の中でシフトJIS文書のファイルサイズが最も小さくなる．

3. モード方式による日本語文書圧縮の欠点

非モード式符号化方式のシフトJISコードに2バイト系モードを導入して短縮表現する圧縮法が文献[4]で提案されている．2バイト文字が最低3文字以上連続する部分の開始位置と終了位置にシフトイン・シフトアウト（0xFF）を挿入し，モードを2バイト系に切り替える．2バイト系モード内では辞書登録文字のみを1バイトの辞書項目で表現する．簡単な圧縮例を示す．

（圧縮前） 22byte

「シフトJISと2バイト文字」

8356 8374 8367 4A 49 53 82C6 32 836F 8343 8367 95B6 8E9A

（圧縮後） 20byte

FF 00 01 02 FF 4A 49 53 82C6 32 FF D86F D843 02 03 04 FF

この例では，辞書項目0x00, 01, 02, 03, 04に2バイト文字の"シ", "フ", "ト", "文", "字"が登録されていると想定している．シフトイン・シフトアウトで囲まれた部分に2バイトの非辞書登録文字が出現する場合には第1バイト目を辞書項目と区別するため表2の変換規則を適用する．

この方法には以下の欠点がある．まず，1バイト文字が出現する毎にモード切り替えが発生し，シフトイン・シフトアウトが随所に挿入される．更に，元文書の文字列情報がシフトイン・シフトアウトで分断される可能性もある．これらの欠点はすべてシフトJISコードが本来，非モード方式であるにも拘らず，モード方式を導入して短縮表現をしているからである．

表2 非辞書登録文字の第1バイト変換規則

81→D6	8D→E0	97→EA	E1→F4
82→D7	8E→E1	98→EB	E2→F5
83→D8	8F→E2	99→EC	E3→F6
84→D9	90→E3	9A→ED	E4→F7
85→DA	91→E4	9B→EE	E5→F8
88→DB	92→E5	9C→EF	E6→F9
89→DC	93→E6	9D→F0	E7→FA
8A→DD	94→E7	9E→F1	E8→FB
8B→DE	95→E8	9F→F2	E9→FC
8C→DF	96→E9	E0→F3	EA→FD

表3 提案法における各コードの用途

用　途	コード値
辞書項目	00~D5
非辞書登録2バイト文字の第1バイト	D6~FD
連続スペース開始指標	FE
仮想2バイト文字の第1バイト	FF

この方法で生成される圧縮ファイルにシフトイン・シフトアウトがどの程度含まれるかについては次節で定量的に論ずる．ここでは，文献[4]で取り上げられている20種の日本語文書について，平均で4.88%，最大で7%以上がシフトイン・シフトアウトで埋められていることだけを付記しておく．

4. 辞書配列と非モード方式による日本語文書圧縮

シフトJIS文書に1バイト文字と2バイト文字が混在できるのは表1にあるように2バイト文字の第1バイト目を0x81~0x9Fと0xE0~0xEFに限定しているからである．それ以外のコードはすべて1バイト文字と判断される．シフトJISコードは大雑把にいえば半角文字は1バイトで，全角文字は2バイトで表現する符号体系である．提案法はこの体系を辞書登録文字は1バイトで，非辞書登録文字は2バイトで表現する符号体系に変換する．1バイト文字と2バイト文字の定義を変更しただけなので提案法の符号化方式は非モード式符号化方式のままである．

1バイト256種のコードを提案法では以下の用途に使用する．0x00~0xD5を辞書項目に，0xD6~0xFDを2バイト文字の第1バイト変換規則に，0xFEを連続スペース開始指標に，そして0xFFを仮想2バイト化に割り当てる．一覧にまとめたのが表3である．

まず，2.で述べた0xFFをすべての1バイト文字の直

前に付加して，1バイト文字を仮想2バイト化する．これでシフトJIS文書を構成するすべての文字を2バイト文字として取り扱うことができる．同時に1バイト文字と2バイト文字の混在という問題も解消する．本来の全角2バイト文字と仮想2バイト文字を合わせて総2バイト文字と称することにする．但し，改行（0x0D0A）だけは事前に全角2バイト文字の範疇に入れておき，（0xFF0D 0xFF0A）とはしない．

総2バイト文字から高出現頻度の上位214文字を辞書に登録する．辞書登録文字には表3の辞書項目0x00(0)~0xD5(213)を付与する．残った総2バイト文字が非辞書登録文字となる．辞書は一次元配列で実装でき，辞書項目は配列添え字で参照できる．

辞書登録文字と非辞書登録文字の選別ができたので，元文書の先頭から順に1文字ずつ読み込み，辞書登録文字ならば辞書項目を1バイトで圧縮ファイルに書き出す．非辞書登録文字ならば第1バイト目を表2の変換規則に従って変換して書き出した後，第2バイト目を無変換のまま書き出す．

変換規則の必要性を簡単に説明する．元文書の先頭文字が非辞書登録文字「あ」で，辞書項目0x82に「い」が登録されていると仮定する．「あ」のシフトJISコードは0x82A0である．もしそのまま「あ」のコードを圧縮ファイルに書き出すと，復元時に先頭の0x82は辞書項目と解釈されてしまう．結果，辞書登録文字「い」が復元されてしまい，正しく「あ」と復元されない．理由は表3で0x00~D5が辞書項目として割り当てられているからである．従って非辞書登録文字の第1バイト目を辞書項目範囲を避けて変換する必要がある．

さて，表4は日本語文書で使用される1バイト文字種と2バイト文字種の数とその容量である．日本語文書は文献[4]で取り上げられたTeX文書10種とオンラインマニュアル10種の計20ファイルである．

表4から日本語文書に出現する1バイト文字は経験的に80種程度，2バイト文字は450種程度と考えてよい．辞書登録の対象は高出現頻度の最大214文字に限定される．しかし，表5が示すように，その214文字種で文書全体の9割以上を捕捉できる．表5の括弧内数字は表4同様，占有バイト数を示している．

提案法を3.の例題「シフトJISと2バイト文字」に適用すると以下のようになる．

（圧縮前） 22byte
8356 8374 8367 4A 49 53 82C6 32 836F 8343 8367 95B6 8E9A
（仮想2バイト化） 26byte
8356 8374 8367 FF4A FF49 FF53 82C6 FF32 836F 8343 8367 95B6 8E9A
（圧縮後） 17byte
03 04 05 00 01 02 D7C6 FF32 D86F D843 05 06 07

この例では，辞書項目0x00, 01, 02, 03, 04, 05, 06, 07に文字"J", "I", "S", "シ", "フ", "ト", "文", "字"

表4 日本語文書における文字種と容量

ファイル名	サイズ	1バイト文字種	2バイト文字種
fax.tex	416byte	26種(118)	77種(298)
floppy.tex	1112	41(540)	46(572)
jrule.tex	2382	61(1576)	85(806)
user.tex	3447	54(483)	243(2964)
dic.tex	6548	39(916)	425(5632)
info.tex	7520	62(1162)	397(6358)
vmap.doc	11624	91(4990)	374(6634)
vz16.doc	14994	88(3776)	401(11218)
dviprt.tex	22837	87(7935)	427(14902)
jtex.tex	26621	86(6791)	528(19830)
lha.doc	27059	82(6785)	576(20274)
ieicej.tex	33039	93(14637)	502(18402)
diet144.doc	33320	79(3378)	556(29942)
mskanji.txt	35831	85(14985)	414(20846)
spcast.doc	56839	82(30365)	483(26474)
manual.tex	57795	89(13483)	592(44312)
siori.txt	61590	77(17208)	874(44382)
dosrefer.txt	119384	91(34434)	699(84950)
hsb.doc	124504	102(54704)	694(69800)
util.doc	164269	98(87773)	563(76496)
平　均	40557	76(15302)	448(25255)

括弧内は占有バイト数

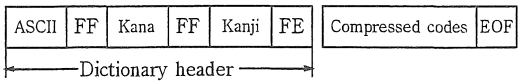

図1 圧縮ファイルのファイル構造

が登録されていると想定している．非辞書登録文字の第1バイトには表2の変換規則を適用している．

提案法をテキストベースの表組みで多用される連続スペースにも対応させる．連長（Run-length）圧縮を応用する．スペースには半角スペース（0x20）と全角スペース（0x8140）が存在する．両者が混在しても，完全に元文書が復元できるよう符号化する．連続数の下限を2，上限を129とする．連続スペース開始指標は0xFEである．0xFEの直後に1バイトで連続数を書き出す．全角連続スペースは半角連続スペースと区別するために連続数に0x80（128）を加算する．

例えば，半角スペースが連続5個出現する場合は，0xFEに続いて0x03（=5-2），すなわち0xFE03を圧縮ファイルに書き出す．全角スペースが連続5個出現する場合は，同様に0xFEに続いて0x83（=5-2+128），つまり0xFE83を書き出す．

図1は提案法で短縮表現した圧縮ファイルの構造である．図中のEOFはファイル終端である．圧縮ファイルの先頭には出現頻度の高い上位214種の文字が辞書

表5 日本語文書における辞書登録文字の占有率

ファイル名	1バイト文字種	2バイト文字種	占有率
fax.tex	26種(118)	77種(298)	100.0%
floppy.tex	41(540)	46(572)	100.0
jrule.tex	61(1576)	85(806)	100.0
user.tex	46(475)	168(2814)	95.42
dic.tex	25(897)	189(5038)	90.64
info.tex	43(1128)	171(5650)	90.13
vmap.doc	70(4954)	144(5830)	92.77
vz16.doc	57(3671)	157(9950)	90.84
dviprt.tex	74(7887)	140(13330)	92.91
jtex.tex	53(6667)	161(17730)	91.65
lha.doc	65(6684)	149(17848)	90.66
ieicej.tex	71(14521)	143(16156)	92.85
diet144.doc	38(3196)	176(27418)	91.88
mskanji.txt	63(14840)	151(19222)	95.06
spcast.doc	44(30187)	170(23968)	95.28
manual.tex	67(13211)	147(39460)	91.13
siori.txt	40(16833)	174(34902)	84.00
dosrefer.txt	61(33887)	153(75328)	91.48
hsb.doc	67(54297)	147(60820)	92.46
util.doc	76(87484)	138(69274)	95.43

占有率とは辞書登録文字が文書全体に占める割合

に登録される．効率を考えて先頭辞書部（Dictionary header）を3つに区分する．第1区分に半角英数文字（ASCII文字）を1バイトで登録する．第2区分に仮名・片仮名・全角英数文字・記号の一部をやはり1バイトで登録する（表6参照）．0xFEと0xFFの2領域は辞書の構造上使用できない．第3区分に漢字などそれ以外の文字を2バイトで登録する．0xFEを辞書終端指標とし，それ以降の圧縮コード部（Compressed codes）に入力文書ファイルの情報を書き出す．

入力文字から辞書項目へのコード変換は探索時間$O(1)$のハッシュ法で実現できる．圧縮時間の短縮を最優先させるためである．

辞書配列$dictionary[i][j](0 \leq i \leq FF, 0 \leq j \leq FF)$は初期値$NIL$で事前に初期化しておく．$dictionary$がハッシュ表に相当する．ここで，$NIL$は辞書サイズ（214）に1を加算した番兵である．

辞書登録された文字$C_i(0 \leq i \leq M)$の第1バイト目をC_{i1}，第2バイト目をC_{i2}とする．仮想2バイト文字の第1バイト目は0xFFとする．$Ci1$，C_{i2}がハッシュ値に対応する．Mは辞書サイズ214以下の登録文字数である．文字C_iの辞書項目をmとすると，ハッシュ表のバケット値は$dictionary[C_{i1}][C_{i2}] = m$と設定できる．これで入力文字の辞書登録の有無とコード変換が探索時間$O(1)$のハッシュ法で実現できる．

コード0xFFを3.のシフトイン・シフトアウトに利用する場合と提案した4.の仮想2バイト文字化に利用する場合との比較が表7である．明らかに提案法の方が出現頻度が低くなっている．

表6 1バイトで辞書登録する文字セット

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	＋	－	±	×	÷	＝
1	ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク	グ
2	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	ダ
3	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ	バ
4	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ	ム
5	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ	
6	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ	（	）	［	］	｛	｝	「	」	ぁ
7	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く	ぐ	け
8	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た	だ	ち
9	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は	ば	ぱ
A	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み	む	め
B	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ	ゐ	ゑ
C	を	ん	改	SP	、	。	，	．	・	：	；	？	！	ー	￥	〒
D	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
E	Q	R	S	T	U	V	W	a	b	c	d	e	f	g	h	i
F	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

改：改行コード，SP：2バイト全角スペース

表7 圧縮ファイルに占める0xFFの割合

ファイル名	シフトイン・アウト		仮想2バイト文字化	
	0xFF	出現率	0xFF	出現率
fax.tex	9 byte	2.37%	7 byte	1.85%
floppy.tex	63	6.27	7	0.75
jrule.tex	93	4.49	10	0.52
user.tex	67	2.82	11	0.47
dic.tex	295	6.33	16	0.37
info.tex	191	3.71	37	0.74
vmap.doc	203	2.39	38	0.46
vz16.doc	603	5.59	114	1.10
dviprt.tex	865	5.05	50	0.31
jtex.tex	599	3.22	131	0.72
lha.doc	1077	6.23	103	0.62
ieicej.tex	1071	4.03	126	0.50
diet144.doc	371	1.91	188	0.96
mskanji.txt	1511	6.71	150	0.71
spcast.doc	1497	5.89	156	0.65
manual.tex	2199	5.51	230	0.60
siori.txt	3265	7.38	377	0.90
dosrefer.txt	3461	4.62	509	0.70
hsb.doc	5049	6.49	410	0.56
util.doc	5581	6.61	292	0.37

5. 性能評価と考察

4.の提案法を以下SSJT(Shotened Shift-JIS Text)と記すことにする．提案法は頻度を利用する典型的な2パス方式の圧縮法である．性能評価のために文献[4]後半の日本語文書圧縮（以下DicJTと略記），さらに以下の3つの既存圧縮法と比較した．2パス方式の代表格である静的ハフマン圧縮（以下Huffと略記）[1]，国産アーカイバのLHA Ver.2.55，インターネット上の標準圧縮ツールgzip Ver.1.2.4の3者である．LHAの実行オプションはcmn2，gzipはfNq9とした．実行環境はNEC PC-98（i486DX2, 66MHz）で統一した．評価に用いた日本語文書は表4の20ファイルである．

各圧縮法による圧縮率を表8に示す．圧縮率は元文書ファイルと圧縮ファイルのサイズの比を%で表記してある．SSJTはDicJTの圧縮率を3~4%改善できることを確認した．また，Huffよりは格段に良い結果を得た．しかし，このままではLHAやgzipには及ばない．

そこで，提案法を前処理として利用した場合の結果を表9に示す．前処理として利用できる理由はSSJTやDicJTがbyte-to-bitの圧縮ではなくbyte-to-byteの圧縮を採用しているからである．前処理を施せばLHAやgzipの単独使用時の圧縮率を更に改善できる．

前処理がDicJTの場合と比較すると元文書が10KB以上でSSJTの優位性が目立つ．しかし，表8のSSJTとDicJTの前処理結果の差がそのまま後段での圧縮率の差に反映されない．これはLHAやgzipが文字列単位のLempel-Ziv圧縮[2]をするからである．つまり，DicJTのモード切り替え文字をそのまま文字列と認識して圧縮している．その結果，前段での圧縮率3~4%の優位は後段で0.5%程度となる．

各圧縮法のプログラムサイズと実行時間を表10に示す．圧縮/復元時間共に元ファイルを10KBに正規化し，その平均時間で示してある．比較を容易にするための措置である．SSJT，DicJT共にハッシュ探索を利用している．SSJTはJIS X 0208の6879字に限定してハッシュ表を用意したためプログラムサイズが小さくなっている．圧縮時間がDicJTの1.3倍となってしまうのは次の2点が主な原因である．第1に，提案法のSSJTは1バイト文字も辞書登録の対象となるため，2バイト文字の登録数が減少する．その結果，第1バイト変換規則の適用回数が増加してしまう．第2に，SSJTの辞書登録対象文字は全文字種である．DicJTの登録対象は2バイト文字のみである．このため出現頻度を基に上位214文字種を選択する際の整列時間が増大する結果となった．

圧縮時には頻度計算，辞書作成，コード変換の3過程が必要となる．復元時には頻度計算の過程が不要であるため高速復元可能である．

表8 各圧縮法による圧縮率

ファイル名	SSJT	DicJT	Huff	LHA	gzip
fax.tex	90.9	91.1	195.0	81.0	81.0
floppy.tex	83.8	90.4	116.7	29.6	28.4
jrule.tex	80.8	86.9	89.0	37.2	36.5
user.tex	67.5	68.9	89.3	50.5	50.4
dic.tex	66.7	71.1	82.0	50.2	50.1
info.tex	66.9	68.5	83.4	48.5	48.4
vmap.doc	71.7	73.0	79.4	43.7	43.4
vz16.doc	69.3	72.0	75.6	36.2	35.7
dviprt.tex	71.6	74.9	81.0	39.3	38.5
jtex.tex	68.3	69.8	79.6	39.2	37.7
lha.doc	60.9	63.8	75.0	40.1	38.7
ieicej.tex	76.7	80.4	80.2	35.5	33.8
diet144.doc	58.5	58.3	74.8	35.3	33.3
mskanji.txt	59.0	62.8	71.5	24.7	22.4
spcast.doc	42.1	44.7	56.2	17.3	15.9
manual.tex	66.4	69.0	77.9	35.5	32.1
siori.txt	67.9	71.8	79.3	36.7	35.2
dosrefer.txt	61.1	62.8	72.6	32.8	29.7
hsb.doc	59.3	62.5	69.9	31.4	28.3
util.doc	48.4	51.4	60.8	24.9	22.9

圧縮率の単位はすべて%

表9 提案法と既存圧縮法を組み合わせた圧縮率

ファイル名	サイズ	前処理：SSJT		前処理：DicJT	
		LHA	gzip	LHA	gzip
fax.tex	416byte	88.7	81.7	84.1	79.3
floppy.tex	1112	35.2	32.4	32.8	30.7
jrule.tex	2382	40.6	40.0	40.2	39.6
user.tex	3447	50.0	49.6	49.1	48.8
dic.tex	6548	46.0	45.8	46.4	46.2
info.tex	7520	46.6	46.4	46.4	46.1
vmap.doc	11624	42.3	42.2	42.0	41.9
vz16.doc	14994	34.1	33.9	34.4	34.2
dviprt.tex	22837	36.4	36.0	37.1	36.6
jtex.tex	26621	35.3	34.9	35.3	35.0
lha.doc	27059	36.4	35.9	36.6	36.1
ieicej.tex	33039	33.2	32.3	33.8	33.0
diet144.doc	33320	30.5	30.0	30.4	29.9
mskanji.txt	35831	21.8	21.4	22.3	21.9
spcast.doc	56839	15.1	14.4	15.3	14.6
manual.tex	57795	30.8	29.4	31.3	29.9
siori.txt	61590	33.7	32.8	34.0	33.2
dosrefer.txt	119384	28.4	26.9	28.9	27.4
hsb.doc	124504	27.7	25.9	28.2	26.5
util.doc	164269	21.3	20.3	21.6	20.8

表10 各圧縮法のプログラムサイズと実行時間

圧縮方法	プログラムサイズ	圧縮時間	復元時間
SSJT	81,619 byte	0.246秒	0.060秒
DicJT	141,155	0.187	0.058
Huff	17,233	0.292	0.330
LHA	36,796	0.141	0.027
gzip	39,910	0.169	0.077

6. おわりに

シフトJIS文書を非モード方式のままで圧縮する方法を提案した．半角文字を1バイト，全角文字を2バイトで表現するシフトJISコードから辞書登録文字を1バイト，非辞書登録文字を2バイトで符号化するコード体系へと変換することで圧縮を実現した．

提案法はbyte-to-byteの圧縮をするため，元文書のバイト列情報を極力保存する．既存圧縮法の前処理に利用して，圧縮率を更に改善することもできた．

謝　辞

本研究は財団法人 日東学術振興財団の第14回（平成9年度）研究助成により達成された．ここに謝意を表する．

参考文献

1) 植松友彦:「文書データ圧縮アルゴリズム入門」，CQ出版社，東京，1994.

2) T.C.Bell, J.G.Cleary and I.H.Witten: Text compression, Prentice Hall, New Jersey, 1990.

3) 荻原剛志:“仮名文字の短縮表現を用いた日本語文書の圧縮について,” 信学論（A），Vol.J74-A, No.9, pp.1431–1438, Sept. 1991.

4) 伊藤 雅，佐藤泰司:“シフトJISコード体系における日本語文書圧縮,” 信学論（A），Vol.J80-A, No.9, pp.1579–1583, Sept. 1997.

5) 日本工業標準調査会:「7ビット及び8ビットの2バイト情報交換用符号化漢字集合 JIS X 0208」，日本規格協会，1997.

6) 日本工業標準調査会:「情報交換用漢字符号—補助漢字 JIS X 0212」，日本規格協会，1990.

7) 日本工業標準調査会:「国際符号化文字集合—第1部 体系及び基本多言語面 JIS X 0221」，日本規格協会，1995.

8) K.Lunde: Understanding Japanese information processing, O'Reilly&Associates,Inc., Sebastopol, 1993.

（受理　平成11年3月20日）