7E－4

# 圧縮された日本語テキストのための パターン照合機械の設計

宮崎哲司＊ 平田博明＊ 古賀寿憲＊＊ 深町修一＊ 有村博紀＊ 篠原武＊

＊九州工業大学　＊＊寿殿

## 1　はじめに

情報検索の問題の1つとして，パターン照合問題がある．パターン照合問題とは，パターンおよびテキストという2つの文字列が与えられたとき，テキスト中におけるパターンのすべての出現を見つけだす問題である．このパターン照合問題を解くアルゴリズムに，Aho-Corasick法 [1] がある．Aho-Corasick法の特長は，パターンよりパターン照合機械という一種の有限オートマトンを作り，そのパターン照合機械をテキスト上で1回走査させるだけで複数のパターンを同時に検出できることである．

Aho-Corasick法を用いて実際に検索を行う際に問題になるのが，ディスクからテキストを読み込むのに要する時間である．この問題に対して，テキストをあらかじめ圧縮しておき，その圧縮されたテキストを復号せずに検索する方法が提案されている [3]．

日本語テキストには，ひらがな，カタカナ，漢字，記号といった複数の文字種が存在し，テキスト中では同じ文字種が連続して出現するという性質がある．この性質を利用して，文字種ごとに圧縮符号を割り当てることにする．このとき，異なる文字種間で同じ符号が割り当てられる可能性があるので，文字種を保持する必要がある．文字種の変わり目に文字種が変わることを示すシフトコードを用いる．この方法により，文字種を考慮に入れずに全体で圧縮符号を割り当てた場合より圧縮率はよくなる．

本稿では，Aho-Corasick法のパターン照合機械を拡張し，シフトコード付き圧縮符号によって圧縮された日本語テキストのためのパターン照合機械を設計した．これを用いて日本語テキストを対象として実験を行った結果，圧縮率は69.7%，走査時間は78.3%に短縮できた．

## 2　日本語テキストの圧縮

圧縮符号としては，テキスト圧縮に最適な符号として知られているHuffman符号 [5] を用いる．Huffman符号とは，テキスト中にある各文字の出現確率によって最適な可変長符号を割り当てた符号である．これは，出現確率の高いものは符号が短くなり，低いものは符号が長くなるという特徴をもつ．

日本語テキスト中の文字は，大きく分けると，1バイト(半角)と2バイト(全角)に分けることができる．2バイト文字はさらに細かく，記号，英数字，ひらがな，カタカナ，ギリシャ，ロシア，漢字の7種類に分けることができる．以上のようにして分けた文字種ごとに，Huffman符号を割り当てる．しかし，異なる文字種の間で同じ符号が割り当てられる可能性があるので，圧縮テキストを見たときどの文字種の符号であるかわからない．そこで，文字種の変わり目にシフトコードを用いて，文字種を識別できるようにする．

シフトコードを用いたHuffman符号について説明する．“知能はAI”というテキストがあるとする．文字種の変わり目にシフトコードという架空の文字があるとすると，“知能→は→AI”となる．このテキストの中の’→’がシフトコードである．この例では，漢字からひらがな，ひらがなからASCIIの2種類が現れている．シフトコードも文字として扱い，Huffman符号を割り当てる．文字種は，漢字からひらがなのときは漢字，ひらがなからASCIIのときはひらがなとする．表1にHuffman符号の例を示す．

表 1: Huffman符号表

| ASCII | 符号 | ひらがな | 符号 | 漢字 | 符号 |
|---|---|---|---|---|---|
| A | 0 | は | 0 | 知 | 0 |
| I | 10 | を | 10 | 能 | 10 |
| ASCII →漢字 | 110 | ひらがな →ASCII | 110 | 漢字 →ひらがな | 110 |
| ASCII →ひらがな | 111 | ひらがな →漢字 | 111 | 漢字 →ASCII | 111 |

この表から，“知能はAI”を符号化すると，“110010 1100110010”となる．ただし，テキストはASCIIから始まると考えているので，先頭にASCIIから漢字へのシフトコードを付け加えている．

## 3　圧縮テキストのためのパターン照合機械

Aho-Corasick法で用いられるパターン照合機械とは，一種の有限オートマトンであり，3つの関数Goto, Failure, Outputで構成される．Goto関数は，現在の状態と入力記号から次状態を決定する．Failure関数は，入力記号に対してGoto関数が定義されていないときに遷移先を決定する．Output関数は，現在の状態で検出されているパターンを表す．このパターン照合機械をシフトコード付きHuffman符号により圧縮された日本語テキスト用に拡張する．

ここでは，表1のHuffman符号表を用いて，パターンの集合 $\Pi = \{$知能は, AIを$\}$ が与えられたときの拡張したパターン照合機械の構成法を示す．まず，与えられたパターンをテキストと同様に符号化する．パターン“知能は”, “AIを”を変換すると，それぞれ“110010110 0”, “01011110”となる．

次にGoto関数を作成する．今回の圧縮法では，同じHuffman符号で複数の文字を表す可能性があるので，パターン照合機械でテキストを走査するときにどの文字かを判別できるようにしなければならない．まず，初期状態1からなるGotoグラフを作り，これをASCIIに

---

[0] Efficient string pattern matching for compressed Japanese text

Tetsuji Miyazaki†, Hiroaki Hirata†, Hisanori Koga††
Shuichi Fukamachi†, Hiroki Arimura†, Takeshi Shinohara†
†Kyushu Institute of Technology, ††Kotobuki-den

対応させる．次にASCII以外の文字種の数だけ，新しい状態を用意し，文字種をそれぞれ割り当てる．ここでは，ひらがなに状態2，漢字に状態3を割り当てる．初期状態1よりASCIIから各文字種へのシフトコードにしたがい，ラベル付きの辺を各文字種に対応する状態まで伸ばしてGotoグラフを作成する．これにより，図1のようなGotoグラフができる．あとは，Aho-Corasick法のGoto関数と同様の作業で作成できる．また，Huffman符号は可変長符号なので，Failure関数の作成にAho-Corasickの手法を直接用いると誤検出が生じる．それを回避するため，それぞれの文字種のHuffman木をGotoグラフに付け加える．Huffman木の葉の部分は，文字の場合は自分の文字種に対応する状態(ASCII: 状態1, ひらがな: 状態2, 漢字: 状態3)，シフトコードの場合は変更後の文字種に対応する状態に設定する．このとき付け加えるHuffman木において，パターン照合機械が必要とするのは，各符号の長さだけである．このことに着目してノード数を減らす方法が考えられている[4]．

最後にFailure関数を作成する．まず，空のキューを用意して，初期状態1をキューに入れる．各文字種に割り当てた状態からのFailure関数の値をそれぞれのHuffman木のルートに設定する．残りの状態に対するFailure, Output関数はAho-Corasick法とまったく同様に作成する．以上のようにして作成されたパターン照合機械が図2である．実線はGoto関数，破線はFailure関数，下線付きの文字列はその状態における出力を表している．

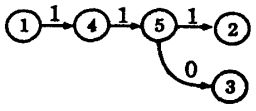

図 1: シフトコードによるGotoグラフ

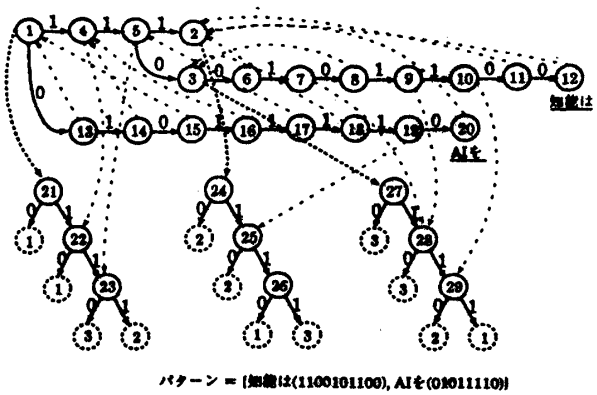


図 2: パターン照合機械

## 4 実験結果

テキストは九州工業大学附属図書館情報工学部分館の蔵書目録データとし，計算機はSPARC station 20を使用して実験を行った．

圧縮前のテキストサイズは7.8Mbyteである．Huffman符号の符号単位を4ビットで圧縮を行った結果，圧縮率は69.7%となった．

圧縮していないテキストを文字符号分割法[2]を用いて走査した時間と圧縮した場合の走査時間を測定した．その結果を表2に示す．この結果より，走査時間は78.3%に短縮できたことがわかる．

表 2: 走査時間

| | データサイズ (Mbyte) | 走査時間 (sec) |
|---|---|---|
| 圧縮前 | 7.8 | 3.23 |
| 圧縮後 | 5.3 | 2.53 |

## 5 考察

今回の実験では，Huffman符号の符号単位は4ビットとした．この場合，1バイトにつき状態遷移と出力の検出がそれぞれ2回必要となる．これに対して，文字符号分割法の場合は1バイトにつき2回の状態遷移と1回の出力の検出でよい．このため走査時間がテキストの圧縮率まで短縮できなかったと考えられる．

次に，Huffman符号の符号単位について述べる．圧縮率だけを考えると符号単位を小さくした方が圧縮率はよくなる．実際，今回のテキストを対象にして符号単位を1ビットで圧縮を行うと，圧縮率は約64.4%となる．しかし，検索を行う場合には1バイトにつき8回の状態遷移と出力の検出をしなければならない．符号単位を4ビットにすると上で述べたように1バイトにつき2回の状態遷移と出力の検出を行えばよいので，多少圧縮率は悪くなるが走査時間はこちらのほうが短くなる．このような最適な符号単位はテキストのエントロピーから決めることができる[3]．今回用いたデータのエントロピーは5.45(bit/文字)であるので，符号単位を4ビットとした．エントロピーの小さいテキストに対しては，符号単位を大きくすると圧縮率が悪くなるので，より小さな符号単位にしなければならない．

## 参考文献

[1] Aho, A.V. and Corasick, M.J.: Efficient string matching: An aid to bibliographic search, Comm. ACM, vol.18,no.6,pp.333-340(1975).

[2] 有川, 篠原: 文字列パターン照合アルゴリズム, コンピュータソフトウェア, vol.4,no.2,pp.2-23(1987).

[3] 深町, 篠原, 竹田: 可変長符号圧縮データのための文字列パターン照合 ーゲノム情報の高速検索技法ー, 1992年情報学シンポジウム,pp.95-103,(1992).

[4] 平田: パターン照合機械のためのハフマン木の縮退, 九州工業大学情報工学部知能情報工学科卒業論文, (1995).

[5] Huffman, D.A.: A method for the construction of minimum-redundancy codes, Proc.IRE 40,pp.1098-1101,(1952).