データベースシステム 125－40
（２００１．７．１８）

HTML 形式の表構造に対する一索引化手法

獅々堀 正幹　岩口 義広　鄭 珉洙　青江 順一
徳島大学　工学部　知能情報工学科
E-mail : { bori, iwaguchi, minsoo, aoe }@is.tokushima-u.ac.jp

WWW 空間上の HTML 文書には，形式的な情報を容易に伝達するために，数多くの表が含まれており，行列方向の単語間の関係や単語の意味情報など，非常に有益な情報を含んでいる．本研究では，これら表構造から固有名詞が有する意味的多義性に関する情報を抽出することを目的としており，その第一段階として，本稿では HTML 形式の表構造内の関係を保持したまま各項目を効率的に索引化する手法を提案する．本手法は，複雑な表にも適用できるセグメンテーション法により，表内での各項目の位置情報をコンパクトなビット列で表現する．また，本手法で表現した位置情報は，奇数ビットが行方向，偶数ビットが列方向の関係を表すため，行列方向の位置関係を高速に照合できる．WWW 空間上から収集した 200 個の表構造（4,836 個の項目数）に対して，表内の座標を索引化する手法と比較実験を行った結果，本手法による索引が 87%コンパクトであり，また，各項目間の照合速度に関しては，本手法が約 5.4 倍高速であり，表構造が複雑になる程，本手法の方が有効であった．

キーワード：表構造解析，索引化，HTML 文書，ネットサーチエンジン，情報抽出

An Indexing Method for Table Structures of HTML Format

Masami SHISHIBORI, Yoshihiro IWAGUCHI, Minsoo JUNG and Jun-ichi AOE
Dpt. of Information Science & Intelligent Systems, Faculty of Engineering, Tokushima University
E-mail : { bori, iwaguchi, minsoo, aoe }@is.tokushima-u.ac.jp

HTML documents in the WWW space frequently include the table structure, which has a very useful information, such as the meanings and relations of words in the table. In this paper, we propose the method to construct the index which keeps the relations in the table structure of HTML format. This method represents the position of each item in the table structure as the compact bit stream. Moreover, since the odd bits of this bit stream show the row relation of each item, on the other hand, the even bits are the column relation, it is very easy and quickly to compare the relation of positions of items in the table. From the experiment result using 200 HTML table structures, which are collected from WWW space by hand, it was found that this method can generate 87% percent smaller index and compare the position relations 5.4 times faster than the indexing method storing the row and column coordinates of each item.

Keywords: Table structure analysis, Indexing, HTML document, Internet searching engine, Information extraction

1．はじめに

近年のインターネット技術の発展は目覚ましく，WWW 空間上には莫大な数の情報が蓄積されるようになった．WWW 空間上に存在する HTML 文書には，形式的な情報を容易に伝達するために，表が頻繁に掲載される．特に，広告ページには，数多くの表が掲載されているという実験報告もされている[1]．これら表構造内には，行・列方向の項目間に関係があり，また，各行・列毎に違った意味を持っている．そして，その意味情報は各行・列の最上位の項目から判定可能である[2]．このように，表構造内には，情報検索や情報抽出の分野に有益な情報を含んでいる．本研究では，これら表構造から固有名詞が有する意味的多義性に関する情報を抽出することを目的としている．

本研究と同様，表構造内から情報を抽出する研究も数件報告されている．伊藤ら[3]は，表構造からオントロジーを構築する手法を提案している．また，吉田ら[2]は，表構造内での単語の出現位置と出現頻度から，表のクラスタリングを行う手法を提案している．

両手法とも，HTML 形式の表を対象としているが，複雑な表構造[1]は扱っていない．一方，嶋田ら[4]は，複雑な表構造を上位列項目，上位行項目，データ項目から成る３重組のリスト形式で表現し，表内から特徴的なデータを抽出する手法を提案している．しかし，データ構造の有効性については論じられておらず，大規模なデータに対しては，空間効率，及び時間効率の面で問題が生じることが予測される．よって，複雑な表構造に対する有効な索引化が実現すれば，上記の研究成果も大きく向上すると思われる．

そこで，本稿では，WWW 空間上に存在する表構造から固有名詞に関する情報抽出を行う第一段階として，複雑な表構造に対して，表内の関係を保持したまま各項目を索引化する手法を提案する．

WWW 空間上に存在する莫大な数の表構造を索引化するためには，表内の各項目の位置を表す情報（位置情報）がコンパクトで，かつ，各項目間の位置関係を高速に判定できるデータ構造でなければならない．この点に関して，本手法では，画像処理の分野で利用されている完全２分木によるセグメンテーション法[5]を応用する．この方法は，縦→横の順番で平面を再帰的に分割し，縦の分割では左側の領域にビット値 0 を右側の領域には１を割り振る．また，横分割の場合には，上側の領域にビット値 0 を下側の領域には１を割り振る．この方法で生成した位置情報は，奇数ビットが行方向，偶数ビットが列方向の関係を表すため，奇数番目のビット値を反転させると行方向で最上位項目の位置情報に対応し，また，偶数番目のビット値を反転させると列方向で最上位項目の位置情報に対応することになり，行列方向の位置関係を高速に判定できる．しかし，従来のセグメンテーション法は，$2^N \times 2^N$ の領域の位置情報をビット列で表現するものであり，WWW 空間上に存在する複雑な表構造には対応できない．そこで，より複雑な表構造に適用できるように，分割の順番，ビット値の設定方法等の考え方を変更した新しい表構造解析アルゴリズムを提案する．

以下，2.では，固有名詞の意味的多義性を解消する上で，表構造解析の必要性，有効性について述べる．次に，3.において，従来の表構造解析手法の概要，及び問題点を明確化した後，4.にて，本手法による位置情報生成法，及び，位置関係の照合方法を説明する．そして，5.で，本手法の評価を行い，最後に，6.で今後の課題について触れる．

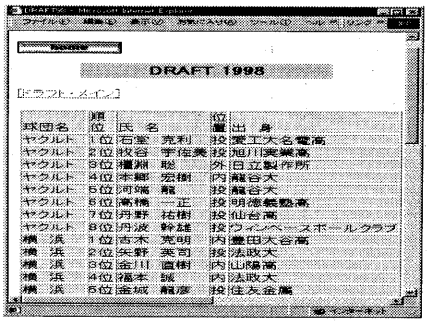
2．表構造解析の必要性

本章では，現在の情報検索システム，特に，ネットサーチエンジンに対し，固有名詞の意味的多義性がもたらす問題点を示し，その問題点に対して表構造解析がどのように利用できるかについて論じる．

一般に，ネットサーチエンジンでは，「組織名」，「人名」，「専門用語」等の固有名詞が検索語として指定されることが多い．しかし，固有名詞には，表記は同じでも異なる意味（意味的多義性）を持つものが多数存在する[6]．そのため，固有名詞が持つ意味的多義性に気付かずに検索すると，従来のネットサーチエンジンでは，検索結果に多くのノイズが含まれ，検索精度を悪化させる原因となる．例えば，「ヤクルト」という固有名詞を検索語として指定すると，「組織（球団）名」の意味を持つ「ヤクルト」と「製品（ドリンク）名」の意味を持つ「ヤクルト」が検索されてしまう．また，検索語に検索意図にあった意味を付与する際に，事前に構築済みのシソーラスを利用することが考えられる．しかしながら，動的な WWW 空間上の単語に静的なシソーラス上の意味情報を付与することは適切とは言えず，WWW 空間上から単語の意味情報を何だかの方法で取得する必要がある[7]．

この問題点に対して，我々は，表構造内に存在する固有名詞の意味情報が各行・列の上位方向の項目から判定可能と考え，WWW 空間上に多数存在する表構造を解析することで，固有名詞の意味情報を自動抽出することを研究課題としている．例えば，図１は，固有名詞“ヤクルト”が項目に含まれる表である．これらを表構造解析し，項目“ヤクルト”の最上位項目を抽出することにより，“ヤクルト”が「球団名」と「企業名」の意味を持つことが分かる．このように，固有名詞の意味情報を WWW 空間上から収集するためには，表構造内の関係を保持したまま各項目を索引化する必要がある．更に，各項目の位置情報は，行列方向の位置関係が高速に判定可能で，かつ，コンパクト性に優れたデータ構造で表現しなければならない．

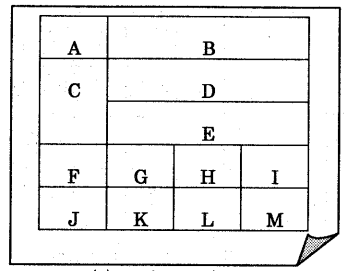
[1] 複数の"rowspan", "colspan"を含む表構造を意味する．



(a)「球団名」の意味を持つ“ヤクルト”の表

(b)「企業名」の意味を持つ“ヤクルト”の表

図1　“ヤクルト”に関する表構造



(a)　ブラウザ画面

```
<html><body><table border = 1>
<tr><td>A</td><td colspan = 3>B</td></tr>
<tr><td rowspan = 2>C</td><td colspan = 3>D</td></tr>
<tr><td colspan = 3>E</td></tr>
<tr><td>F</td><td>G</td><td>H</td><td>I</td></tr>
<tr><td>J</td><td>K</td><td>L</td><td>M</td></tr>
</table></body></html>
```

(b)　HTMLファイル

図2　ブラウザ上の表とそのHTMLファイルの例

3．位置情報の表現方法

HTML形式の表の項目サイズや行数・列数は，表に関するタグ情報（td, tr, rowspan, colspan等）から取得できる．例えば，図2(a)の複雑な表に対しては，図2(b)に示す“rowspan”，“colspan”タグから，各要素の大きさを知ることができる．本章では，タグ情報から表内の位置情報を表現する従来法として，座標を用いる手法とリストを用いる手法を取り上げ，各手法の問題点を明らかにした後，本手法で用いるセグメンテーション法について説明する．

3．1　座標による表現方法

表構造は，縦×横の長方形を細分化したものであるので，これを二次元配列で考えると，座標に置き換えることができる．この方法は，データ構造やアルゴリズムが単純であり，整列された表には有効であるが，複雑な表に対しては，問題点が生じる．例えば，図2(a)の表を座標に置き換えたものを図3に示す．

図3に示す座標表現では，項目B，C，D，Eのように“colspan”，“rowspan”を含む場合，項目C

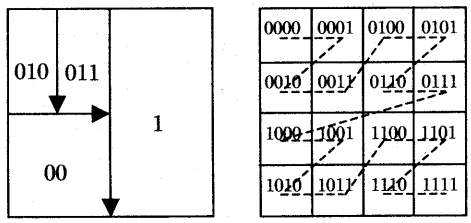
(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)
(3,0)	(3,1)	(3,2)	(3,3)
(4,0)	(4,1)	(4,2)	(4,3)

図3　表の座標表現例

では2個，B，D，Eでは3個分の座標をとり，無駄な領域を確保することになる．また，項目Bに属する下位の項目を検索する場合，まず，Bの座標(0,1),(0,2),(0,3)を調べ，次に，これら3個の座標に対して下位の座標をそれぞれ検索しなければならない．このように，2個以上の座標を含む場合，その個数分に比例して検索コストが増大する．

3．2　リストによる表現方法

リストで位置情報を表現する方法は，表の最上位項目から下位項目にノードをつなげていく方法である．検索を行う際，ノードをたどることにより高速に検索できる．しかし，表構造内の位置関係を表現するためには，1項目に対して最低4つのアークが必要である．また，各項目情報はどの位置に存在するか，情報をもたせておく必要がある．このように，記憶容量に問題が生じると容易に想像できる．そこで，位置情報を効率的に表現する方法を次節で説明する．



(a) 完全2分木による表現　(b) Z-ordering による表現

図4　セグメンテーション法による表現方法

3．3　セグメンテーション法による表現方法

本手法では，画像処理分野で利用されている完全2分木によるセグメンテーション法を適用する．この方法は，縦→横の順番で領域を再帰的に分割し，位置情報（ビット列）を生成する．例えば，図4に示すように縦の分割の場合，分割軸の左側の領域には‘0’を加え，右側に‘1’を加える．横の分割の場合は，上側に‘1’，下側に‘0’を加える．この位置情報の表現方法は，多次元検索の分野ではz-ordering としても有名であり，$2^N \times 2^N$ の整列された領域の位置情報をビット列で表現するものである．

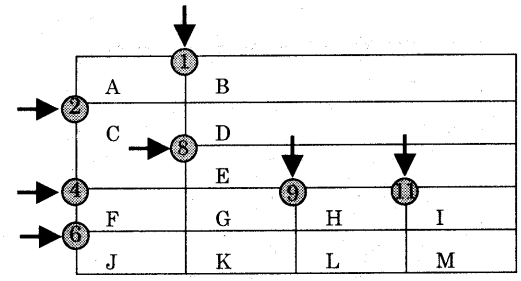
しかし，表構造はより複雑なため，上記手法をそのまま適用することはできない．そこで，本稿では，上記手法での分割軸の選び方，及びビット値の生成方法の考え方を変更し，表構造解析に適した新しいセグメンテーション法を提案する．

4．本手法による表構造解析手法

4．1　位置情報生成法の概要

位置情報の生成方法について，図5の表を例として，説明する．まず，表内には多数の分割軸が存在するが，各分割軸の始点をカッティングポイントと呼ぶ．図5(a)の表では，①⑨⑪が縦方向の分割軸に対するカッティングポイント，②④⑥⑧が横方向のものである．尚，カッティングポイントの特定方法の詳細アルゴリズムは，4．3．1で述べる．

本手法では，2つの制約事項に従って，適用すべきカッティングポイントの順番を制御し，縦→横の順に表を分割しながらビット列を生成する．まず，最初の制約事項を以下に示す．



(a) 表構造の例

00	10		
0110	11101010		
	11101011		
011110	111110110	111110111110	111110111111
011111	111111110	111111111110	111111111111

(b) 位置情報生成例

図5　表構造と位置情報の生成例

【制約事項1】

より外側で，原点よりに位置するカッティングポイントから分割を行う．

より外側から分割するのは，複数の小さな表が集まって複雑な表を形成していると考え，最初は，できる限り大きな表に分割しながら，順次，表を細分化するためである．また，原点よりから順次分割するのは，同じ行（列）に属する項目の出現順を正しく位置情報に反映させるためである．図5(a)の表において，縦方向のカッティングポイントは，①→⑨→⑪という順番で選択され，横方向は，②→④→⑥→⑧となる．つまり，①の後に②で分割された結果，図5(b)のように，項目Aの位置情報（“00”）と項目Bの位置情報（“10”）が確定する．尚，カッティングポイントの選択方法の詳細アルゴリズムは，4．3．2，位置情報の確定判定方法は，4．3．3で述べる．

次に，2番目の制約事項を以下に示す．

【制約事項2】

適切なカッティングポイントが存在しない場合は，位置情報が未確定な全ての領域にダミービット値‘1’を与える．

尚，上記の制約で，「適切でない」とされるカッティングポイントは，下記の条件を満たすものである．

【条件1】カッティングポイントの上位に，位置情報が未確定の領域が存在する．

【条件2】カッティングポイントが存在しない．

まず，条件1が適合するのは，②の後に⑨を選択した場合である．ここで，⑨の上位に存在する項目D，Eの位置情報は未確定であるため，縦方向で適切なカッティングポイントは存在しないと判定し，項目A，B以外の全ての位置情報に‘1’（3ビット目）を与える．その後，④，⑥，それぞれの後に⑨を選択しようとするが，同様に，ダミービットが5，7ビット目に与えられる．また，⑨の後に横方向のカッティングポイントを探すが，横方向は全て適用済みであり，条件2が満たされ，未確定領域（項目H，I，L，M）の10ビット目に‘1’を付与する．

4．2　位置情報による項目検索

本手法で生成した位置情報は，奇数ビットが横方向の位置関係を，偶数ビットは縦方向の位置関係を表す．この縦・横の位置情報を表すビットを操作することにより，各位置関係による項目を容易に検索できる．

例えば，図5(b)の表で項目Aの縦方向での下位項目を知りたい場合，まず，項目Aの位置情報（“00”）を得た後，縦方向の位置関係を表す偶数ビットを（下位を表す）‘1’に反転する．そして，“01”を接頭辞にもつ位置情報を検索することにより，項目C,F,Jが得られる．

また，項目Lの縦方向での最上位項目を検索する場合，まず，項目Lの位置情報（“11111111110”）を得た後，偶数ビットの内でビット値が‘1’である部分を（上位を表す）‘0’に反転（“10101010100”）する．そして，そのビット列で最長一致検索を行うと“10”が得られ，項目Bが検索される．

最後に，項目Bと縦方向，項目Fと横方向に関係のある項目を検索する場合，項目B（“10”），項目F（“011110”）の位置情報を得る．その後，項目Bでは，縦方向での検索となるので，奇数ビットの位置情報を抽出し，項目Fでは，横方向での検索となるので偶数ビットの位置情報を抽出する．そして，それぞれのビット値を組み合わせたビット列“11*[2] 1*0”で接頭辞検索を行うと項目G,H,Iが得られる．

[2] *は，‘0’あるいは‘1’のビット値を表す．

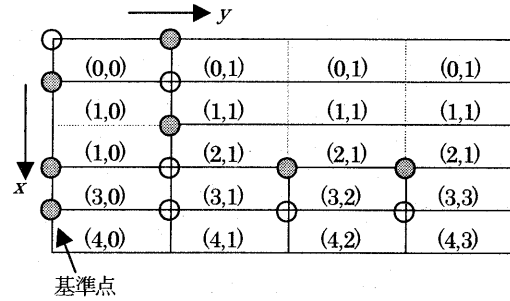


図6　図5(a)の表構造に対する中間表現

このように，簡単なビット操作を施すだけで，一意に目的とする項目を検索することができる．尚，本検索手法では，2種類のデータベースが必要である．一方は，項目名から位置情報を得る索引，他方は，位置情報から項目名を検索するものである．また，位置情報が‘0’，‘1’の2値であること，接頭辞検索，最長一致検索が必要であることを考慮し，本手法で生成した位置情報を2進木パトリシアトライ構造[8]に格納する．

4．3　表構造解析アルゴリズム

表構造解析を行う上で，タグ情報から必要なデータを取り出し，中間表現として最小単位の縦×横の二次元配列を用意する．また，表構造をそのままの形式で表現するために，図6(a)のように各項目の左上の点を基準として座標を置く．この点を，基準点と呼ぶ．また，項目が2個以上の配列要素を含む場合は，各要素にその項目の基準点を置く．ここで，二次元配列の各要素は図6(b)のデータ構造を持つ．

4．3．1　カッティングポイントの取得方法

上記の中間表現を用いて，基準点の中からカッティングポイント（$CP=(CPx,CPy)$）を取得する．カッティングポイントか否かの判定は，各基準点の行・列上位方向どちらか一方にでも他のカッティングポイントが存在していない場合に，その基準点をカッティングポイントとする．この処理は，各項目に対して，上位方向に隣接する項目の基準点の座標を調べることで実現する．

4．3．2　カッティングポイントの選択方法

適切なカッティングポイントを選択するため，縦・横方向のカッティングポイントを個別に格納する2つのキューを用いる．このキューを用いたカッティングポイントの選択方法を図5(a)の表を例に説明する．

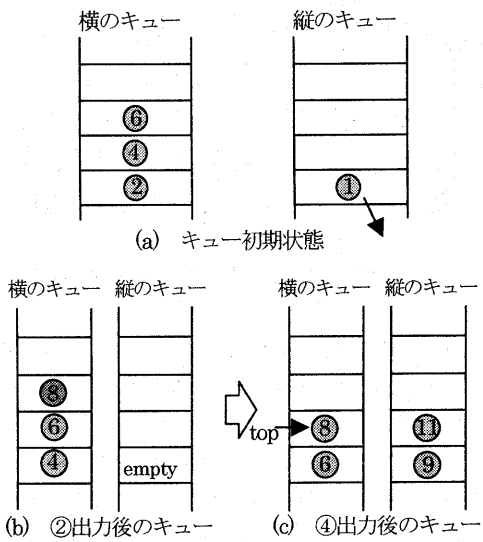


図8 カッティングポイントのキューによる制御

まず，初期設定として表内の外枠に存在するカッティングポイントをそれぞれ図8(a)に示すようにキューに格納する．そして，縦→横の順にカッティングポイントを対応するキューから出力し，ビット生成を行う．ここで，処理を行った分割軸上にカッティングポイントが存在すればキューに格納する．例えば，①による分割軸上には，⑧が存在するため，縦キューから①が出力された後には，⑧が横キューに入力される．

次に，①，②を出力した後のキューの状態を図8(b)に示す．ここで，縦キューには空であり，適用すべきカッティングポイントがないため，ダミービットが付与される．この状態は，**4．1の条件2**に対応する．

続いて，横のキューから④を取り出し，位置情報を生成する．ここで，④の分割軸上に⑨，⑪が存在するので順次，縦キューに格納する（図8(c)）．この後，縦キューから⑨を出力する際に，縦キューの *bottom* に位置するカッティングポイント（CP_{btm}）と横キューの *top* に位置するカッティングポイント（CP_{top}）の座標を比較する．これは，キューからカッティングポイントが出力される毎に実行され，処理対象となるカッティングポイントが**4．1の条件1**を満たすか否かを検証するために行われる．この検証は，縦方向のカッティングポイントの場合，CP_{btm} のx座標が CP_{top} のx座標より大きければ，条件1を満たしていると判定できる．また，横方向の場合には，y座標が大きければ条件1が真となる．先の例では，⑨は縦方向であり，⑧より⑨のx座標が大きいため，ダミービットが割り振られる．

4．3．3 位置情報の確定判定方法

生成された位置情報の確定判定には，各項目毎にフラグを設定する．このフラグは，各項目の四方四辺が分割されたか否かを示すものである．即ち，四辺が全て分割された項目は，位置情報が確定したと判断する．手順としては，縦方向のカッティングポイントの場合には，y座標が同じときに，反対に横方向の場合には，x座標が同じときに，フラグを1インクリメントし，値が4になれば確定とする．

5．評価

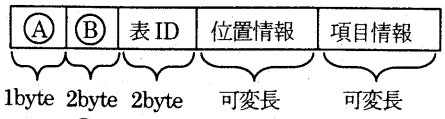
本手法の有効性を確認するため，提案手法による索引の空間効率と時間効率を評価した．また，大規模な表に対する本手法の問題点，及び，その改善策を示す．

5．1 実験データの内容

実験データとして，PCスペック表を中心にWWW空間から200個の表を人手で収集した．この表構造には，総項目数が4,836個，HTMLタグのcolspanが計617個，rowspanが計1,160個含まれていた．表構造の平均サイズは，12行4列，最大行数は63行，最大列数は19列であった．また，これらの表構造データを座標表現による位置情報と提案手法による位置情報との2種類で索引化した．尚，稼動マシンは，OSはWindows 2000，CPU Pentium3/500MHz，メモリ256Mbyteである．

5．2 提案アルゴリズムの検証

本稿で提案した表構造解析アルゴリズムの正当性（出力結果が正しいか否か）を実験的に検証する．実験を行うために，位置情報をビット列で表現した索引（本手法）と座標で表現した索引，2種類の索引を準備する．検証方法は，収集した表に含まれる全ての項目間の関係を検索し，提案手法による検索結果が，座標表現による検索結果と同じである場合，正解と判定する．つまり，座標による検索を正解とみなし，提案手法による検索結果の正解率を求める．項目間の関係の検索方法は，表内の全ての項目に対して，縦方向と横方向の下位項目検索を行う．実験の結果，100%の正解率が得られ，本アルゴリズムの正当性が実証された．



Ⓐ: 項目情報のバイト数

Ⓑ: 位置情報のビット数

図9　データベースのフォーマット

表1　提案手法と座標表現の記憶容量

	提案手法(Kbyte)	座標表現(Kbyte)
Bucket	480	542
Index	152	185
Total	632	727

5．3　精度的評価

5．2で用いた座標により位置情報を表現する手法を比較手法とし，本手法の空間効率，及び時間効率を評価する．空間効率の評価方法は，本手法（ビット列）による位置情報と座標表現による位置情報をパトリシアトライ構造に登録し，そのサイズを比較した．尚，データベースのフォーマットを図9に示す．本手法によるデータベースは，図9の位置情報の部分がセグメントにより生成されたビット列となり，一方，比較手法では，位置情報の部分は各項目の座標値を持つことになる．また，位置関係による項目検索を実現するために，項目をキーとするものと位置情報をキーにする，2種類のデータベースから構成される．

まず，各索引のサイズを比較した実験結果を表1に示す．空間効率において，双方ともほとんど変わらないが，本手法の方が，座標のデータベースよりも 87% コンパクトな容量となった．

次に，時間効率においては，**5．2**で行った検索時間を本手法と座標表現による手法で比較した．実験結果を図10に示す．時間効率においては，本手法が縦方向の下位項目検索で7.4倍，横方向の下位項目検索では2.9倍高速であった．総合的にみても5.4倍高速であることが分かる．特に，縦方向の下位項目検索時間に大きな差が見られる．一般に，座標検索では，下位の項目数が多くなると，データベースへのアクセス回数が増加する．WWW 上の表構造の特徴は，列数が少なく，行数が長いという特徴があるため，座標による検索の場合，項目数に比例して検索時間も大きくなる．一方，本手法では，下位項目検索の際に，ビット列上での接頭辞検索を行えるため，時間の増大を押さえられる．

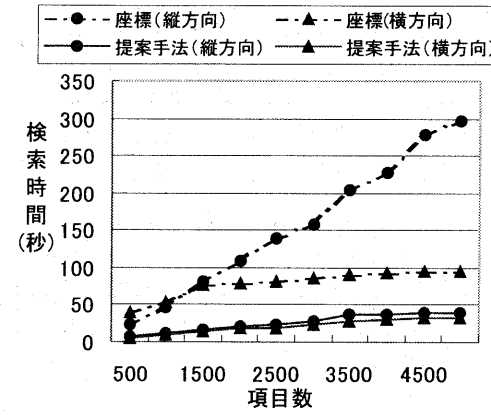


図10　提案手法と座標表現の検索時間

5．4　表構造の複雑度と検索時間との関係

表の構造が複雑になるに従って，本手法の有効性がどの程度向上するかを確認するため，表を複雑度の度合いから分類し，各クラスの表に対して検索時間を測定した．まず，実験データの表構造を以下の基準に従って，3つのクラスに分類した．

【クラス分けの基準】

複雑：表構造中の0行目0列目に一つでも rowspan, colspan が含まれる表；

単純：表構造中に rowspan, colspan は一つも含まず，整列された表；

その他：表構造中の0行目0列目以外に rowspan, colspan が含まれる表；

この結果，『複雑』には 67 個，『単純』には 108 個，『その他』には 25 個の表が分類された．そして，各クラス毎に関連項目検索を行い，検索時間を測定した．尚，ここで実施した関連項目検索とは，行項目と列項目を1個ずつ入力し，行項目については縦方向の下位項目検索を行い，列項目については横方向の下位項目検索を行って，それらの交わり部分の項目（共通項目）を検索することを意味する．そして，『複雑』と『単純』のクラスに属する各表に対して，0 行目 0 列目の各項目を入力として関連項目検索を実施した場合の検索時間を表2に示す．

表2　表構造の複雑度と検索時間との関係

クラス	提案手法(sec)	座標表現(sec)	倍率
単純	9.46	20.31	2.1
複雑	4.65	29.82	6.4

表3　大規模な表構造に対する検索時間

検索方向	提案手法(sec)	座標表現(sec)	倍率
縦方向下位	820.60	446.55	0.54
横方向下位	12.67	18.51	0.68

表2より，表構造が複雑なほど，比較手法に比べて本手法が有効であることが分かる．これは，rowspanやcolspanを含む項目の関連項目を検索する場合，座標表現では，一項目あたり複数の座標により検索しなければならないが，提案手法では，一項目あたり一つのビット列から関連する複数の項目を一度に検索できることが原因として考えられる．

5．5　大規模な表に対する問題点と改善策

WWW 空間上に存在する大規模な表構造には，ブラウザの表示機能の関係上，横幅に比べて縦の長さが非常に長いという特徴がある．このような大規模な表に対して，本手法による位置情報を作成すると，右下方向の項目になるに従ってビット列が非常に長くなり，また，縦幅と横幅が異なるため，多数のダミービットが付与されてしまう．その結果，表構造が大規模になるに従って，検索コストが悪化することが予測される．

実際の影響を確認するために，大規模な表構造として，徳島大学の組織表を対象にして実験を行った．この組織表は，410行8列から成り，1,687項目を含んでいた．この表の全ての項目に対して，下位項目検索を行った結果（検索時間）を表3に示す．予想通り，ビット列が非常に長くなり（最長816ビット），縦横方向ともに座標表現の検索よりも遅くなった．

この問題点に対して，我々は，大規模な表構造に対して，表構造を水平方向に分割し，分割した表構造毎に位置情報を生成する改善策を提案する．以下，実際の分割方法，位置情報の管理方法について述べる．

まず，表構造の分割により，一つの項目が複数の表に跨って分割されるのを防ぐため，分割行が，項目間の境界部分であることをチェックする必要がある．このチェックは，その行の0列目にカッティングポイントが存在するか否かを調べれば良い．

また，表分割を行った場合，その情報を索引に反映させる必要がある．そこで，図9のフォーマットにおける表IDの先頭1ビットを分割された表か否かを示すフラグとして用いる．更に，分割した表には，それぞれSubIDを持たせ，表IDの後に1byteのSubID情報を付与したフォーマットで索引を形成する．

6．まとめ

本稿では，HTML 形式の表構造を解析し，表内の位置情報をコンパクトで，かつ，項目間の関係が高速に照合可能なビット情報で表現するアルゴリズムを考案した．今後，実際に WWW 空間上から収集した大規模な数の表構造を索引化し，表構造から固有名詞の意味情報を抽出するアルゴリズムを提案し，実験・評価を行う予定である．本改善策は，現在，実装中であり，完成し次第，実験を行う予定である．

参考文献

[1] 井上香織，高橋克巳：検索のための広告文書構造化，情報処理学会大57回全国大会論文集，1V-4，pp.207-208(1998).

[2] 吉田稔，鳥澤健太郎，辻井潤一：表形式からの情報抽出手法，言語処理学会第6回年次大会，pp.252-255(2000).

[3] 伊藤史郎，大谷紀子，上田隆也，池田祐治：属性オントロジーの抽出と統合を用いた実空間と情報空間のナビゲーションシステム，人工知能学会誌，Vol.14，No.6，pp.1001-1009(1999).

[4] 嶋田和孝，遠藤勉：製品情報表からの特徴データの抽出，情報処理学会自然言語処理研究会報告資料，NL99-133，pp.107-113(1999).

[5] 土屋英亮，伊藤秀一，渡辺修史：MDL原理と2分記構造セグメンテーションを用いた画像の無ひずみ符号化アルゴリズム，電子情報通信学会論文誌，Vol.J80-D-II，No.2，pp.415-425(1997).

[6] 西野文人，落合亮：抽出情報の実体あいまい性の解消，言語処理学会第6回年次大会ワークショップ論文集，pp41-48（2000）.

[7] 堀井則彰，上原邦昭：失敗からの知識獲得とテキスト分類に基づくインターネットからの情報収集，情報処理学会論文誌，Vol.41，No.1，pp.24-35（2000）.

[8] M. Shishibori, M. Okada, T. Sumitomo, J. Aoe, Design of a Compact Data Structure for the Patricia Trie, IEICE Transaction on Information and Systems, Vol. E81-D, No.4, pp.364-371 (1998).