数理解析研究所講究録
1489巻 2006年 216-222

繰返し構造をもつラベル付順序木の簡潔な表現法

斉藤智哉 (Tomoya Saito), 中村篤祥 (Atsuyoshi Nakamura), 工藤峰一 (Mineichi Kudo)
北海道大学 (Hokkaido University)

ABSTRACT

本論文では、構造把握を目的とした、ラベル付順序木の繰返し構造を明示的に表す簡潔な表現である**繰返し表現木**を提案する。また、この表現のうち節点数が最も少ないものを求めるという問題を考え、節点数 n の木に対してその問題を $O(n^3)$ 時間で解くアルゴリズムを示す。

1. はじめに

インターネットの普及と発展に伴い、WWW上には大量かつ多様な情報がWebページという形で蓄積されている。これら膨大な情報を効率良く扱うために計算機による処理が必要となる。しかし、これらHTML文書には文書に含まれる情報の意味が記述されていないため、直接このような文書から計算機によって自動的に情報を得ることは困難である。

そこで、Webページに含まれる情報の自動的な抽出や統合を目的とする研究が盛んに行われている [1–3]。

このような研究のアプローチの一つとして、HTML文書に現れる繰返し構造に着目した手法がある [4–6]。たとえば、図1では、製品ひとつに対応する情報が記述された構造が3回繰返されている。このようなHTML文書の繰返し構造を把握することができれば、情報抽出や情報統合などへのさまざまな応用が期待できる。

図1 Webページの例

Changらは、HTML文書からタグ列の繰返しパターンを発見し、情報を切り出すためのルールを自動的に生成する手法 [4] を提案した。Liuらは、タグの入れ子構造から得られるtag treeというラベル付順序木に現れる繰返し構造を発見することで、data recordと呼ばれる情報の単位構造を抽出する手法 [5] を提案した。これらが部分的な情報の切り出しを行うのに対し、南野らは、タグ列の繰返し構造を階層的に発見することで、Webページ全体の構造化を行う手法 [6] を提案した。

これらの手法はWebページに関する経験則などを用いて繰返し構造の把握を行っており、アプリケーションに特化した手法となっている。

本研究では、ラベル付順序木の繰返し構造を明示的に表す簡潔な表現である**繰返し表現木**を提案し、節点数最小の繰返し表現木を求めることにより繰返し構造を把握する手法を提案する。繰返し表現木は繰返し構造をコンパクトにまとめた表現であるため、節点数が少ないものほどより尤もらしい繰返し構造の解釈であると考えられる。本手法は経験則を用いないため、Webページに限らずラベル付順序木として表すことのできる他の対象にも広く適用することができる。本論文では、この問題を木の節点数 n に対して時間計算量 $O(n^3)$ で解くアルゴリズムを示す。実験によると、実際のHTML文書に対して本手法を適用することで、ブラウザ上での目視により繰返し構造と解釈された部分のうち約80%が把握できた。

2. 問題設定

2.1. 繰返し表現木

根付き木で、同じ親をもつすべての子の節点間に全順序が与えられた木を順序木という。木を図示するとき、子は左から右に順序付けされているとする。本論文では、全ての節点にラベルが与えられたラベル付順序木を考える。

連続的な繰返し構造をもつラベル付順序木をコンパクトに表現するために、**繰返し情報節点**というものが導入されたラベル付順序木を考える。**繰返し情報節点** v_1 とは、繰返し回数 r（2以上）でラベル付けされた節点であり、v_1 の順序付けされ

たすべての子 $c_1, c_2, ..., c_k$ を根とする部分木列が v_1 の親 p の子を根とする部分木列として r 回繰返された木を表現しているものとする (図 2 参照)。繰返し情報節点 v を、それを用いずにそれが表現

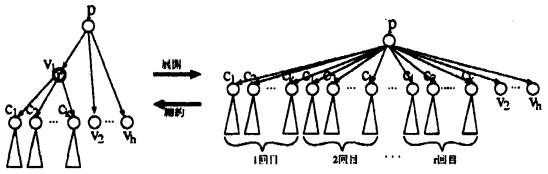


図 2 繰返し情報節点とその展開

している木に変換する操作を v の**展開**といい、その逆の操作を**縮約**という。根と葉以外の節点として繰返し情報節点を 0 個以上もつ木を**繰返し表現木**という。繰返し表現木 R に含まれるすべての繰返し情報節点を展開すると、繰返し情報節点を 1 つも含まない普通のラベル付順序木 T が得られる。このとき、「繰返し表現木 R は、ラベル付き順序木 T を表現する」という。

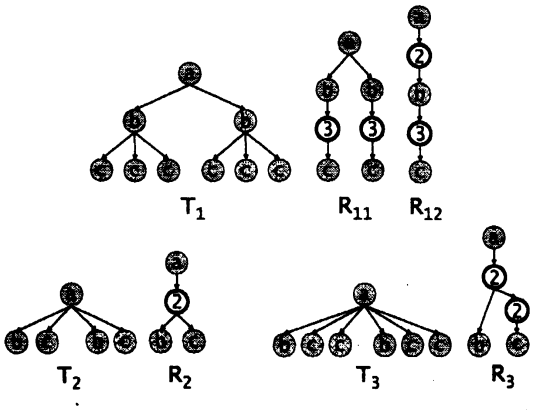


図 3 繰返し表現木の例

例 1 図 3 の T_1, T_2, T_3 は一般的なラベル付順序木であり、それ自身を表現する繰返し表現木でもある。R_{11} と R_{12} は T_1 を、R_2 は T_2 を、R_3 は T_3 を表現する繰返し表現木である。

2.2. 最小繰返し表現木算出問題

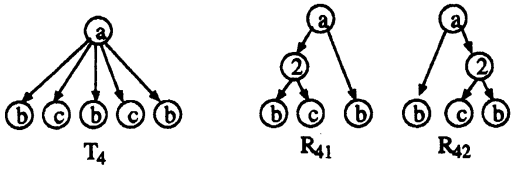
本論文では、以下のような問題を扱う。

問題 1 (節点数最小繰返し表現木算出問題) 与えられたラベル付順序木 T を表現する節点数最小の繰返し表現木 R を求めよ。

ただし、ここでいう節点数とは、繰返し情報節点を含めたすべての節点の数のことである。

例 2 図 3 のラベル付順序木 T_1, T_2, T_3 の場合、それらを表す節点数最小のラベル付順序木はそれぞれ R_{12}, R_2, R_3 である。このとき節点数はそれぞれ、$9 \to 5, 5 \to 4, 7 \to 5$ というように減っており、よりコンパクトな表現になっている。

注意 1 節点数最小の繰返し表現木は一意に定まらない場合がある。例えば、下図の R_{41} と R_{42} は共にラベル付き順序木 T_4 を表現する節点数最小の繰返し表現木である。このように節点数最小の繰返し表現木が複数存在する場合には、それらの内 1 つを求めればよいことにする。



3.2. 節で説明するように、最小繰返し表現木算出問題は、以下のように定義される「高さ 1 のラベル付順序木に対する節点重み和最小の繰返し表現木算出問題」を繰返し解く問題に帰着できる。

問題 2 (高さ 1 のラベル付順序木に対する節点重み和最小の繰返し表現木算出問題) ラベルの集合を Σ とし、Σ の各要素 l には実数重み $w(l)$ が与えられているものとする。T を Σ 上のラベル付順序木で高さ 1 の任意の木であるとする。このとき、T を表現する節点重み和最小の繰返し表現木 R を求めよ。ただし、繰返し情報節点の重みは 1 とする。

3. アルゴリズム

この節では、まず、高さ 1 のラベル付順序木に対する 節点重み和最小の繰返し表現木算出問題を解くアルゴリズムを示し、その後、そのアルゴリズムを使った節点数最小繰返し表現木算出問題を解くアルゴリズムを示す。

3.1. 高さ 1 のラベル付順序木に対する 節点重み和最小の繰返し表現木算出問題を解くアルゴリズム

まず、記述を簡略化するため、以下のような表記法を導入する。高さが 1 で葉の数が n のラベル付き順序木 T に対し、T の子を i 番目から j 番目までに限った木を $T[i, j]$ で表す。同じラベル l の根をもつ木 R_1 と R_2 に対し、$R_1 + R_2$ は R_1 を前 (左) にして 2 つの木の根を重ねてできる木と

する。つまり、木 R_1+R_2 は、R_1 の深さ1の節点を根とする部分木列の後に R_2 の深さ1の節点を根とする部分木列を加えたものを、深さ1の節点を根とする部分木の列としてもつ木である(図4右から2番目の木参照)。任意の自然数 h と繰返し表現木 R に対し、$h\times R$ は、R の根と深さ1の節点列の間にラベル h の繰返し情報節点を挿入した形の木を表すものとする。つまり、$h\times R$ は、深さ1の節点としてラベル h の繰返し情報節点のみをもち、R の深さ1の節点を根とする部分木列を,深さ2の節点を根とする部分木列としてもつ木である(図4右端の木参照)。

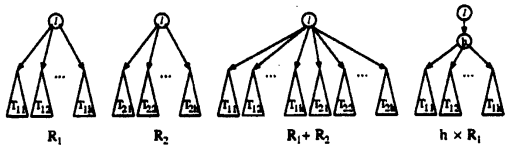


図4 根のラベルが等しい2つの繰返し表現木の足し算と繰返し表現木のスカラー倍

命題 1 繰返し表現木 Rが、高さ1で葉の数が $n\geq 2$ のラベル付き順序木 T を表現する節点重み和最小繰返し表現木であれば、以下の *Case 1* または *Case 2* のどちらかが成り立つ。ただし、$R[i,j]$ は $T[i,j]$ を表現する節点重み和最小繰返し表現木とする。

Case 1 $kh=n$ を満たすある自然数 k と h が存在し、$R=h\times R[1,k]$ を満たす。

Case 2 $1\leq k<n$ を満たすある k が存在し、$R=R[1,k]+R[k+1,n]$ を満たす。

(証明) R の深さ1の節点が1つの節点 v のみの場合は、展開したときに葉が2つ以上になるという仮定から v は繰返し情報節点でなければならない。したがって $kh=n$ を満たすある自然数 k と h が存在し、$T[1,k]$ を表現するある繰返し表現木 R' が存在して $R=h\times R'$ と書ける。R は節点重み和最小であるので R' は $T[1,k]$ を表す節点重み和最小の繰返し表現木でなければならない。したがってこのとき、Case 1が成り立っている。

R の深さ1の節点数が2以上の場合を考える。根と1番目の深さ1の節点のみからなる木 R_1 を展開した木を $T[1,k]$、根と2番目以降の深さ1の節点のみからなる木 R_2 を展開した木を $T[k+1,n]$ とする。このとき、R_1 は $T[1,k]$ を表現する繰返し表現木、R_2 は $T[k+1,n]$ を表現する繰返し表現木であり、$R=R_1+R_2$ と書ける。R は節点重み和最小なので、R_1,R_2 はそれぞれ $T[1,k],T[k+1,n]$ を表現する節点重み和最小の繰返し表現木でなければならない。よってこのとき、Case 2が成り立っている。 □

MinH1Tree % [Minimize Height-One Tree]
入力 l: 根節点のラベル、$(l_1,m_1),(l_2,m_2),...,(l_n,m_n)$: 高さ1の木 T の葉の (ラベル, 重み) の列
出力 (N_R,m_R): T を表現する節点重み和最小の繰返し表現木 R の根節点 N_R とその節点重み和 m_R

```
 1: m[i,i] ← m_i for i = 1,2,...,n
 2: m[i,j] ← ∞
      for i = 1,2,...,n, j = 1,2,...,n, i < j
 3: for k = 2 to n do
 4:   for i = 1 to n − k do
 5:     for j = 0 to k − 2 do
 6:       if m[i,i+j] + m[i+j+1,i+k−1]
            < m[i,i+k−1] then
 7:         m[i,i+k−1] ← m[i,i+j]
              +m[i+j+1,i+k−1]
 8:         type[i,i+k−1] ← (2,j)
 9:       end if
10:     end for
11:     l ← 1
12:     while i+lk+k−1 ≤ n do
13:       if (l_{i+lk}, l_{i+lk+1}, ..., l_{i+lk+k−1})
            ≠ (l_i, l_{i+1}, ..., l_{i+k−1}) then
14:         break
15:       end if
16:       if m[i,i+k−1]+1 < m[i,i+lk+k−1]
          then
17:         m[i,i+lk+k−1] ← m[i,i+k−1]+1
18:         type[i,i+lk+k−1] ← (1,k)
19:       end if
20:       l ← l+1
21:     end while
22:   end for
23: end for
24: m_R ← m[1,n]
25: N_R ← ConstructMinTree(l, l_1, ..., l_n, type)
26: return (m_R + 1, N_R)
```

図5 アルゴリズム MinH1Tree

この命題より、T を表現する節点重み和最小の繰返し表現木を求めるには、Case 1か Case 2を満たす候補木 R の中から探せばよいことがわかる。したがって、$j-i<n-1$ に対する $R[i,j]$ が求まっていれば、R を求めることができる。同様に、$j-i<k$ に対する $R[i,j]$ が求まっていれ

ば、$j-i=k$ を満たす $R[i,j]$ を求めることができる。よって、$R[i,i]=T[i,i]$ という事実を使って、動的計画法により T を表現する節点重み和最小の繰返し表現木 R を求めることができる。

図5に、高さ1のラベル付順序木に対する節点重み和最小の繰返し表現木算出問題を解くアルゴリズム MinH1Tree を示す。このアルゴリズムでは、$1 \leq k \leq n-i+1$ を満たす k に対し、すべての i に対する $T[i,i+k-1]$ を表現する節点重み和最小の繰返し表現木 $R[i,i+k-1]$ の根節点を除いた節点重み和 $m[i,i+k-1]$ を $k=1,2,...,n$ の順番に求める。$k=1$ の場合は、$R[i,i]=T[i,i]$ なので $m[i,i]=m_i$ となる(1行目)。$k \geq 2$ の場合は、命題1の2つの Case を満たす候補木の中から節点重み和が最小となるものを探す。Case 2 を満たすものは、5行目から10行目で探される。Case 1 を満たすものは、11行目から21行目で探される。11行目から21行目では、i から k 個単位の $h \geq 2$ 回の繰返しに対して $m[i,i+hk-1]$ を計算しているが、k が小さい場合から計算しているので、$k=k_0$ のときには、すでに i から $k<k_0$ 個単位の繰返しに対して、Case 1 を満たす候補木の探索は終っている。$T[i,i+k_0-1]$ に対しては Case 1 を満たす候補木は他にはないので、$k=k_0$ の処理に入った時点で $m[i,i+k_0-1]$ には、Case 1 を満たす候補木の節点重み和の最小値が入っている。したがって、$k=k_0+1$ の処理を行なうときにはすべての i に対し、$m[i,i+k_0-1]$ には $R[i,i+k_0-1]$ の根節点を除いた節点重み和が設定されている。

したがって、$m[1,n]$ には、$k=n$ のとき、T に対する節点重み和最小の繰返し表現木 $R=R[1,n]$ の、根を除いた節点重み和が設定される。T を表す節点重み最小の繰返し表現木 R は、それぞれの $m[i,i+k-1]$ を計算するときに Case 1 と Case 2 でどちらの方が小さくなるか、またその Case においてどの単位で繰返したり、どの位置でわかれたりすれば最小となるのかを type$[i,i+k-1]$ に記憶させているので、type$[1,n]$ から最小になるように選ばれた繰返し単位や分割位置を辿ることにより節点重み和 $m[1,n]$ を達成する繰返し表現木 R を $O(n)$ の時間で構成できる。このようなアルゴリズムを ConstructMinTree とする。MinH1Tree は $m[1,n]$ を m_R、ConstructMinTree で求まった R の根節点を N_R として (m_R,N_R) を出力する。よって、アルゴリズム MinH1Tree は与えられた節点重み付きラベル付き順序木 T に対し、T を表現する節点重み和最小の繰返し表現木 R の節点重み和 m_R と R の根節点 N_R の組を出力する。

命題 2 アルゴリズム *MinH1Tree* の時間計算量は $O(n^3)$ である。

(証明) 5行目から10行目の for ループは $O(n)$、12行目から21行目までの while ループは $O(n)$ であるので、3行目から23行目までのループは $O(n^3)$ である。他の処理はすべて $O(n)$ であるから、全体として $O(n^3)$ で計算できる。 □

3.2. 最小繰返し表現木算出問題を解くアルゴリズム

この節では、与えられたラベル付順序木 T を表現する節点数が最小の繰返し表現木 R を求めるアルゴリズム Conv2MinRRTree (図6) を説明する。

Conv2MinRRTree % [Convert to the Minimum Repetition Representation Tree]
入力 N_T: ラベル付順序木 T の根節点
出力 (N_R,m_R): T を表現する節点数が最小の
繰返し表現木 R の根節点 N_R とその節点数 m_R

1: **if** N_T が子を持たない **then**
2: return $(N_T,1)$
3: **else**
4: N_T のラベルを l、子の列を $C_1,C_2,...,C_k$ とする。次の STEP 1〜4 を順に実行。
5: STEP1.
$(N_i,m_i) \leftarrow$ Conv2MinRRTree(C_i) for i=1,2,...,k
6: STEP2.
$(l_1,l_2,...,l_k) \leftarrow$ Labeling$(N_1,N_2,...N_k)$
7: STEP3.
$(N_R,m_R) \leftarrow$MinH1Tree$(l,l_1,l_2,...,l_k,m_1,m_2,...,m_k)$
8: STEP4.
根が N_R の木 R の葉節点でラベルが l_i の節点を N_i に置換
9: **end if**
10: return (N_R,m_R)

図6 アルゴリズム Conv2MinRRTree

Conv2MinRRTree は、T の根節点が入力されると、T を表現する最小節点数の繰返し表現木 R の根節点 N_R とその節点数 m_R の組を出力する。根だけからなる高さが0の木 T が与えられた場合は、T を表現する繰返し表現木が T そのものしかないため、$(N_T,1)$ を出力する。

高さが1以上の木 T の根節点 N_T が入力された場合は、N_T の子の列 $C_1, C_2, ..., C_k$ に対応する葉節点 $L_1, L_2, ..., L_k$ をもつ高さ1の木 T' で、それぞれの葉節点が以下の条件 (1) を満たすラベル $l_1, l_2, ..., l_k$ をもつものを求める。ただし、C_i を根とする部分木を T_i とする。

$$l_i = l_j \Leftrightarrow T_i = T_j \text{ for } (i,j) \in \{1,2,...,k\}^2 \quad (1)$$

このようなラベル付けを行なうのが図6の Labeling 関数である (STEP 2)。図6のアルゴリズムでは、Labeling 関数の引数として Conv2MinRRTree (STEP 1) で求めた T_i を表現する最小繰返し表現木 R_i の根節点 N_i の列を渡しているが、Conv2MinRRTree は決定的アルゴリズムなので $T_i = T_j$ と $R_i = R_j$ は等価となることに注意されたい。R_i の節点数は T_i の節点数以下であるため、オーダー的には差がでないが R_i で比較した方が効率がよい。このようにして求めたラベル l_i の実数重み $w(l_i)$ を R_i の節点数 m_i に設定する。また、T' の根節点のラベルは N_T のラベル l とし、l の実数重みは1と設定する。T からこのように構築された高さ1の木 T' とラベルの重み w に対する「高さ1のラベル付順序木に対する節点重み和最小の繰返し表現木算出問題」を関数 MinH1Tree で解き (STEP 3)、得られた T' に対する節点重み和最小の繰返し表現木 R の根節点 N_R とその節点重み和 m_R を得る。最後に R の葉節点各々を、ラベルが l_i であれば T_i を表現する最小繰返し表現木 R_i の根節点に置換する (STEP 4)。Conv2MinRRTree は、こうしてできた R の根節点 N_R と節点重み和 m_R の組を出力する。

命題 3 ラベル付順序木 T に対し、*Conv2MinRRTree* の出力が (N_R, m_R) であったとする。このとき、N_R を根とする繰返し表現木 R は T を表現する節点数最小の繰返し表現木であり、その節点数は m_R である。

(証明) T が高さ0の場合は、T は根節点 N_T のみからなり、この場合 $(N_R, m_R) = (N_T, 1)$ である。このとき、T を表現する節点数最小繰返し表現木も根節点 N_T のみからなる木であり、その節点数は1である。よって T の高さが0の場合、命題は成り立つ。

高さが $k-1$ 以下のラベル付順序木に対して命題が成り立っているとする。T を高さが k の木とする。T の根節点 N_T の子の列を $C_1, C_2, ..., C_k$ とし、それらを根とする T の部分木を $T_1, T_2, ..., T_k$ とする。数学的帰納法の仮定より、Conv2MinRRTree の STEP 2 で求まる (N_i, m_i) は、T_i を表現する節点数最小の繰返し表現木 R_i の根節点 N_i とその節点数 m_i の組になっている。T を表現する節点数最小の繰返し表現木の1つを R^* とする。このとき、R^* において、展開したときに $T_1, T_2, ..., T_k$ になる部分 $R_1^*, R_2^*, ..., R_k^*$ はそれぞれ $T_1, T_2, ..., T_k$ の節点数最小繰返し表現木になっている。したがって $R_1^*, R_2^*, ..., R_k^*$ を $R_1, R_2, ..., R_k$ で置き換えた木 R^{**} も T を表現する節点数最小の繰返し表現木である。この繰返し表現木 R^{**} において、$R_1, R_2, ..., R_k$ の内部以外の繰返し情報節点をすべて展開した木を Q とする。これは Q を縮約することによって節点数最小の繰返し表現木 R^{**} が得られることを意味する。Q において $R_1, R_2, ..., R_k$ をラベル $l_1, l_2, ..., l_k$ の葉節点に置き換えた木 T' は、明らかに MinH1Tree に入力される木に等しい。$R_1, R_2, ..., R_k$ の節点数 $m_1, m_2, ..., m_k$ を、ラベル $l_1, l_2, ..., l_k$ の重みと考え、根節点のラベルの重みを1とみなせば、T' の節点重み和と Q の節点数は等しい。T' に対する縮約と Q に対する縮約は1対1に対応し、T' から縮約により出来る木 T'' からラベル $l_1, l_2, ..., l_k$ の葉節点を $R_1, R_2, ..., R_k$ に置き換えてできる木は、対応する Q の縮約によりできる木 Q' に一致し、T'' の節点重み和と Q' の節点数は常に等しい。したがって T' に対する節点重み和最小の繰返し表現木 R' のラベル $l_1, l_2, ..., l_k$ の葉節点を $R_1, R_2, ..., R_k$ に置き換えて出来る木 R は、Q を縮約して出来る繰返し表現木の中で節点数最小のものであり、その節点数は R' の節点重み和に等しい。STEP 3, 4 ではそのような R に対する根節点 N_R 及び節点数 m_R を求めている。よって、T の高さが k の場合も命題が成り立つ。

□

命題 4 節点数 n の任意のラベル付き順序木に対し、アルゴリズム *Conv2MinRRTree* の時間計算量は $O(n^3)$ である。

(証明) 入力木 T の高さ h に関する数学的帰納法で証明する。

$h = 0$ のとき、2行目でリターンするので、明らかに $O(1)$ である。

$h \geq 1$のとき、$h < h_0$を満たす高さhの木T^*に対するConv2MinRRTreeの計算時間$Time(T^*)$が、ある定数cに対して

$$Time(T^*) \leq cn^3$$

であると仮定する。Tの高さを$h = h_0$とする。Tの深さ1の節点を根とする木を$T_1, T_2, ..., T_k$とする。また、$T_1, T_2, ..., T_k$の節点数を$n_1, n_2, ..., n_k$とする。$T_1, T_2, ..., T_k$は高さがh_0より小さいので、仮定より、$T_i (1 \leq i \leq k)$に対しConv2MinRRTreeの計算時間は高々cn_i^3時間である。よって、Step 1にかかる時間time_1は

$$\text{time}_1 = \sum_{i=1}^{k} Time(T_i) \leq \sum_{i=1}^{k} cn_i^3$$

である。

Step 2のLabeling関数では、$T_1, T_2, ..., T_k$に対する最小繰返し表現木$R_1, R_2, ..., R_k$の全ての組合せに関して比較を行い、同じ木には同じラベルを、異なる木には異なるラベルを付ける。R_iの節点数は高々n_iであり、R_iとR_jの比較はある定数c_2に対して高々$c_2 \min\{n_i, n_j\}$時間でできる。したがって、Step 2にかかる時間time_2は

$$\text{time}_2 \leq c_2 \sum_{i=1}^{k} \sum_{j=i+1}^{k} \min\{n_i, n_j\} \leq c_2 nk$$

である。

Step 3にかかる時間time_3は、命題2よりある定数c_3に対して

$$\text{time}_3 \leq c_3 k^3$$

である。

Step 4にかかる時間time_4は、ある定数c_4に対して

$$\text{time}_4 \leq c_4 k$$

である。したがって、$Time(T)$に対して次式が成り立つ。

$$Time(T) \leq c \sum_{i=1}^{k} n_i^3 + c_2 nk + c_3 k^3 + c_4 k$$

$c \geq c_2, c_3, c_4$とすると、

$$Time(T) \leq c \sum_{i=1}^{k} n_i^3 + cnk + ck^3 + ck$$

$\sum_{i=1}^{k} n_i^3 + nk + k^3 + k \leq n^3$より(後述)、

$$Time(T) \leq cn^3$$

よって、Tの高さが$h = h_0$のときも成り立つので、Conv2MinRRTreeの計算量は$O(n^3)$である。

$\sum_{i=1}^{k} n_i^3 + nk + k^3 + k \leq n^3$を示す。
$n = \sum_{i=1}^{k} n_i + 1$より、

$$\begin{aligned}
\text{右辺} = n^3 &= \left(\sum_{i=1}^{k} n_i + 1\right)^3 \\
&= \left(\sum_{i=1}^{k} n_i\right)^3 + 3\left(\sum_{i=1}^{k} n_i\right)^2 + 3\sum_{i=1}^{k} n_i + 1 \\
&= \sum_{i=1}^{k} n_i^3 + \left(\sum_{i=1}^{k} n_i\right)^3 - \sum_{i=1}^{k} n_i^3 \\
&\quad + 3\left(\sum_{i=1}^{k} n_i\right)^2 + 3\sum_{i=1}^{k} n_i + 1
\end{aligned}$$

ここで、n_iは全て1以上なので、
$\left(\sum_{i=1}^{k} n_i\right)^3 - \sum_{i=1}^{k} n_i^3 \geq k^3 - k$が成り立つ。
よって、

$$n^3 \geq \sum_{i=1}^{k} n_i^3 + k^3 - k + 3\left(\sum_{i=1}^{k} n_i\right)^2 + 3\sum_{i=1}^{k} n_i + 1$$

また、$k \leq n - 1 = \sum_{i=1}^{k} n_i$であるから、

$$n^3 \geq \sum_{i=1}^{k} n_i^3 + k^3 + 3\left(\sum_{i=1}^{k} n_i\right)^2 + 2\sum_{i=1}^{k} n_i + 1$$

一方、

$$\begin{aligned}
\text{左辺} &= \sum_{i=1}^{k} n_i^3 + k^3 + nk + k \\
&\leq \sum_{i=1}^{k} n_i^3 + k^3 + \left(\sum_{i=1}^{k} n_i\right)^2 + 2\sum_{i=1}^{k} n_i
\end{aligned}$$

よって左辺 ≤ 右辺が成り立つ。 □

4. 実験

HTML文書のtag treeを最小繰返し表現木に変換した際、どれほど繰返し構造を把握できるかを評価する実験を行った。

4.1. 方法

データとして、商品の情報が掲載されているWebページを6ページ、検索エンジンの検索結果ページを3ページ用いた。各HTML文書はTidy [7]を用いて整形したのち、tag treeの最小繰返し表現木を求め、元の木の節点数と比較した。また各ページのうち、Webブラウザ上での目視によって繰返し構造とみなせる部分に着目し、最小繰返し表現木においてその部分が繰返しとして縮約できているかを調べた。なお、節点のラベルが同一であるかの比較にはタグの種類のみを用い、タグに付加される属性は用いなかった。また、テキスト部分はその内容によらず同じラベルの節点とみなした。

4.2. 考察

節点数は平均して45.1%に減少した。また、繰返し構造と判断された部分のうち78.6%が繰返し構造として縮約された。楽天を例にとると、繰返し構造とみなした部分がすべて繰返し構造として縮約されており、目視による解釈と一致した構造把握が行えたと言える。また節点数も比較的少なくなり、多くの繰返し構造が縮約されていると考えられる。一方、Yahoo!検索結果では、繰返し構造とみなした部分のうち、繰返し単位を構成する要素数の異なりによって縮約できなかった部分が多かった。

表1 実験結果

Webページ	c_1	c_2	c_3
Yahoo!	1188	701(59.0%)	10/21(47.6%)
Amazon	987	588(59.6%)	5/5(100.0%)
ヨドバシカメラ	1759	634(36.0%)	33/37(89.1%)
TSUKUMO	771	444(57.6%)	2/5(40.0%)
TSUKUMO(2)	2328	864(37.1%)	16/22(72.7%)
楽天	2056	676(32.9%)	30/30(100.0%)
Yahoo!検索	496	317(63.9%)	4/10(40.0%)
google	312	171(54.8%)	8/10(80.0%)
msn検索	334	226(63.1%)	6/9(66.7%)
total	10231	4621(45.1%)	114/145(78.6%)

(注) c_1 : tag treeの節点数
c_2 : 繰返し表現木の節点数 (tag treeの節点数に対する割合)
c_3 : 縮約できた繰返し構造数/目視による繰返し構造数

5. まとめ

本稿では、ラベル付順序木の繰返し構造を簡潔かつ明示的に表す表現を提案した。また、その表現のうち節点数が最小のものを節点数nに対して$O(n^3)$で求めるアルゴリズムを示した。経験則を用いた既存法 [4–6] では類似した構造の繰返しも抽出可能であるが、今回提案した表現では抽出できないという問題点がある。情報抽出の観点からは類似した繰返し構造の把握も必要であり、そのための手法の検討が今後の課題として挙げられる。

参考文献

[1] A. Laender, B. Ribeiro-Net, A. Silva and J. Teixeira, A Brief Survey of Web Data Extraction Tools, *ACM SIGMOD Record*, **31**, 2(2002), 84–93.

[2] 山田泰寛・池田大輔・坂本比呂志・有村博紀, WWWからの情報抽出:ウェブラッパーの自動構築, 人工知能学会論文誌. **19**, 3(2004), 302–310.

[3] N. Kushmerick, Wrapper Induction for Information Extraction. PhD thesis, University of Washington, 1997.

[4] C-H. Chang and S-C. Lui, IEPAD: Information Extraction Based on Pattern Discovery. *Proceedings of the 10th International Conference on the World Wide Web*, 2001, 681–688.

[5] B. Liu, R. Grossman and Y. Zhai, Mining Data Records in Web Pages. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, 601–606.

[6] 南野朋之, 繰返し構造に基づいたWebページの構造化, 情報処理学会論文誌, **45**, 9(2004), 2157–2167.

[7] Tidy, http://www.w3.org/People/Raggett/tidy/